T.C.
GEBZE TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

İKİ PARÇALI ÇİZGELERDE ETKİN ALT-ÇİZGE ARAMASI

MEHMET BURAK KOCA

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

GEBZE
2020

T.C.
GEBZE TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

İKİ PARÇALI ÇİZGELERDE ETKİN ALT-ÇİZGE ARAMASI

MEHMET BURAK KOCA

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

DANIŞMANI
PROF. DR. FATİH ERDOĞAN SEVİLGEN

GEBZE
2020

T.R.

GEBZE TECHNICAL UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

EFFICIENT SUBGRAPH SEARCH IN BIPARTITE GRAPHS

MEHMET BURAK KOCA

A THESIS SUBMITTED FOR THE DEGREE OF

MASTER OF SCIENCE

DEPARTMENT OF COMPUTER ENGINEERING

THESIS SUPERVISOR

PROF. DR. FATİH ERDOĞAN SEVİLGEN

GEBZE

2020

GEBZE TEKNİK ÜNİVERSİTESİ	**YÜKSEK LİSANS JÜRİ ONAY FORMU**

GTÜ Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 03/01/2020 tarih ve 2020/02 sayılı kararıyla oluşturulan jüri tarafından 08/01/2020 tarihinde tez savunma sınavı yapılan Mehmet Burak Koca'nın tez çalışması Bilgisayar Mühendisliği Anabilim Dalında YÜKSEK LİSANS tezi olarak kabul edilmiştir.

JÜRİ

ÜYE (TEZ DANIŞMANI)	: Prof.Dr. Fatih Erdoğan SEVİLGEN
ÜYE	: Doc.Dr.Didem GÖZÜPEK KOCAMAN
ÜYE	: Dr. Öğretim Üyesi Mehmet BAYSAN

ONAY

Gebze Teknik Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ...../...../..... tarih ve ...../..... sayılı kararı.

İMZA/MÜHÜR

Prof. Dr. Ümit DEMİR
Gebze Teknik Üniversitesi
Fen Bilimleri Enstitüsü Müdürü

ÖZET

Alt-çizge eşbiçimlilik probleminin çözümü için 70'li yıllardan beri çalışmalar yapılsa da çizge boyutlarındaki artış sebebiyle yeni yöntemler yayımlanmaya devam etmekte ve problem popülerliğini korumaktadır. Artan çizge boyutları ile baş edebilmek için birçok algoritma çizgelerin yapısal ve anlamsal özelliklerinden faydalanmaktadır fakat sadece birkaçı çizgelerin karakteristik özelliğini hesaba katmaktadır. Genel çizgeleri hedef alan yaklaşımlar tüm çizge türleri üzerinde eşbiçimlilik sorgusu yapabilmekte fakat özel çizgelere has özelliklerin kullanılmasıyla elde edilebilecek faydalardan mahrum kalmaktadırlar.

Özel bir çizge türü olan iki parçalı çizgeler üzerinde alt-çizge eşbiçimlilik problemi açık bir araştırma alanıdır: Bildiğimiz kadarıyla, şimdiye kadar iki parçalı çizgeler için özelleştirilmiş alt-çizge eşbiçimlilik algoritması bulunmamaktadır. Tez çalışması kapsamında, iki parçalı çizgelerin karakteristik özelliklerinden faydalanarak alt-çizge eşbiçimlilik problemini bu çizgelerde daha yüksek performans ile çözebilen özgün birebir ve yaklaşık eşleme algoritmaları geliştirilmiştir.

Bu tez çalışmasında birebir eşleme yapan iki farklı algoritma sunulmaktadır. Algoritmalardan biri dal-ve-sınır yaklaşımına sahiptir. Bu algoritma, iki parçalı yapının sağladığı imkân doğrultusunda güçlü bir budama tekniğine sahiptir ve bu sayede uygunsuz dallanmalar daha oluşmadan budanmaktadır. Diğer birebir eşleme algoritması indeks tabanlı bir yaklaşıma sahiptir. Algoritma iki parçalı çizgelere uygun kazayağı indeks yapısını kullanarak uygunsuz parçadaki indeks ögelerini filtreleme imkânı sağlar. Geliştirilen son algoritma indeks tabanlı algoritmaya ayrıt eksikliği üzerinden esneklik sağlayarak yaklaşık alt-çizge eşleme işlemini yüksek performans ile gerçekleştirmeyi sağlamaktadır. Geliştirilen üç algoritma da literatürde bulunan aynı türdeki yüksek performanslı algoritmalar ile karşılaştırmalı performans testine tabi tutulmuştur. Sonuçlarda, geliştirilen algoritmaların rakiplerine oranla, çizgelerin düğüm ve ayrıt sayısına göre, iki ile bin kat arasında daha iyi performans gösterdiği gözlemlenmiştir.

Anahtar Kelimeler: Alt-çizge Eşbiçimlilik, İki Parçalı Çizgeler, Birebir Eşleme, Yaklaşık Eşleme.

SUMMARY

Although there have been studies since 1970s for the subgraph isomorphism problem, new methods continue to be published because the problem is still popular due to the increase in the graph size. There are a lot of algorithms that exploit structural and semantic attributes of the graphs to handle the increase in the graph size but only a few of them consider graph characteristics. The approaches that aim the general graphs can do subgraph isomorphism search on all graph types, but they can't benefit from obtainable advantages by using idiosyncrasies of some special graphs.

The subgraph isomorphism problem in bipartite graphs, a special graph type, is an open research area. Comprehensive literature search show that, as far as we know, there is no algorithm for the solution of the problem in the bipartite graphs. Within the scope of this thesis study, novel exact and in-exact algorithms with high performance are proposed for the solution of the problem on these type graphs by exploiting graph characteristic of bipartite graphs.

Two exact match algorithms have been proposed in this thesis study. One of the algorithms is based on branch-and-bound approach. The algorithm has a powerful pruning technique thanks to the partitioned structure of bipartite graphs. It can prune infeasible branches before they are generated. Another proposed exact matching algorithm is an index-based solution. The algorithm only indexes necessary data and provides efficient filtering by using triplets as index structure. Triplets are very suitable for the bipartite graphs and they allow to do efficient indexing and filtering. The last algorithm allows performing approximate subgraph isomorphism search in bipartite graphs with high-performance. All the three proposed algorithms have been compared with their state-of-art competitors. The proposed algorithms show two to a thousand-time better run-time performance results for various graph sizes or densities.

Key Words: Subgraph Isomorphism, Bipartite Graphs, Exact Match, In-exact Match.

TEŞEKKÜR

Yüksek lisans eğitimim ve akademik gelişimimde desteğini ve yardımlarını hiçbir zaman esirgemeyip, bu süreçte yargılayıcı değil destekleyici tavrı ve engin bilgisi ile çalışmalarıma olabilecek en iyi şekilde danışmanlık eden saygıdeğer hocam Prof. Dr. Fatih Erdoğan Sevilgen'e,

Bütün çalışmam boyunca yanımda olan, bilgi ve tecrübelerini benimle paylaşan, başta Hadi Alizadeh olmak üzere tüm değerli çalışma arkadaşlarıma ve saygıdeğer hocalarıma,

Son olarak, sadece akademik değil bütün alanlarda göstermiş oldukları ilgi, sevgi ve desteklerden dolayı sevgili eşim İlknur Koca'ya ve tüm değerli aile üyelerime en içten teşekkürlerimi sunarım.

İÇİNDEKİLER

SİMGELER ve KISALTMALAR DİZİNİ

Simgeler ve Kısaltmalar		**Açıklamalar**
GED	:	Çizge Benzetim Uzaklığı
YOL	:	Yeniden Oluşturma Listesi
⋈	:	Girişim İşlemi

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

1. GİRİŞ

Çizgeler, nesneler ve bu nesneler arasındaki ilişkileri içeren veri kümelerini modellemede uzun yıllardan beri yaygın olarak kullanılmaktadır. Bu veri kümeleri genellikle önemli çalışma alanlarına aittir ve değerli bilgiler içermektedir.

Bu çalışma alanlarından biri olan Biyoinformatik alanında araştırmacılar canlılara ait biyolojik verileri analiz ederek anlamlı ve faydalı bilgiler elde etmektedir. Biyoinformatik alanındaki bu veri kümeleri genellikle biyolojik nesneler ve bunlar arasındaki etkileşimleri içerdiği için araştırmacılar bu veri kümelerini analiz ederken çizge modellemesinden sıklıkla faydalanmaktadır.

Enfeksiyon hastalıklarına sebep olan patojen proteinleri ile taşıyıcı insan proteinleri arasındaki etkileşimler ağını (İng. PHI Network) içeren veri kümesi Biyoinformatik alanındaki popüler veri kümelerinden biridir. Bu veri kümeleri, patojen proteinleri ile etki ettiği taşıyıcı insan proteinleri arasındaki etkileşimleri içermektedir.

Bir patojen etki ettiği insandan faydalanmak için proteinleri ile etkileşim kurmaktadır. Bu etkileşimler sebebiyle insan proteinlerinin eksik, fazla ya da farklı işlevsellik göstererek görevlerini yerine getirmesine engel olmaktadır. Araştırmacılar bu etkileri analiz ederek patojenlerin sebep olduğu protein etkileşimlerinin arasında yapısal bir benzerlik olup olmadığının araştırılmaktadır.

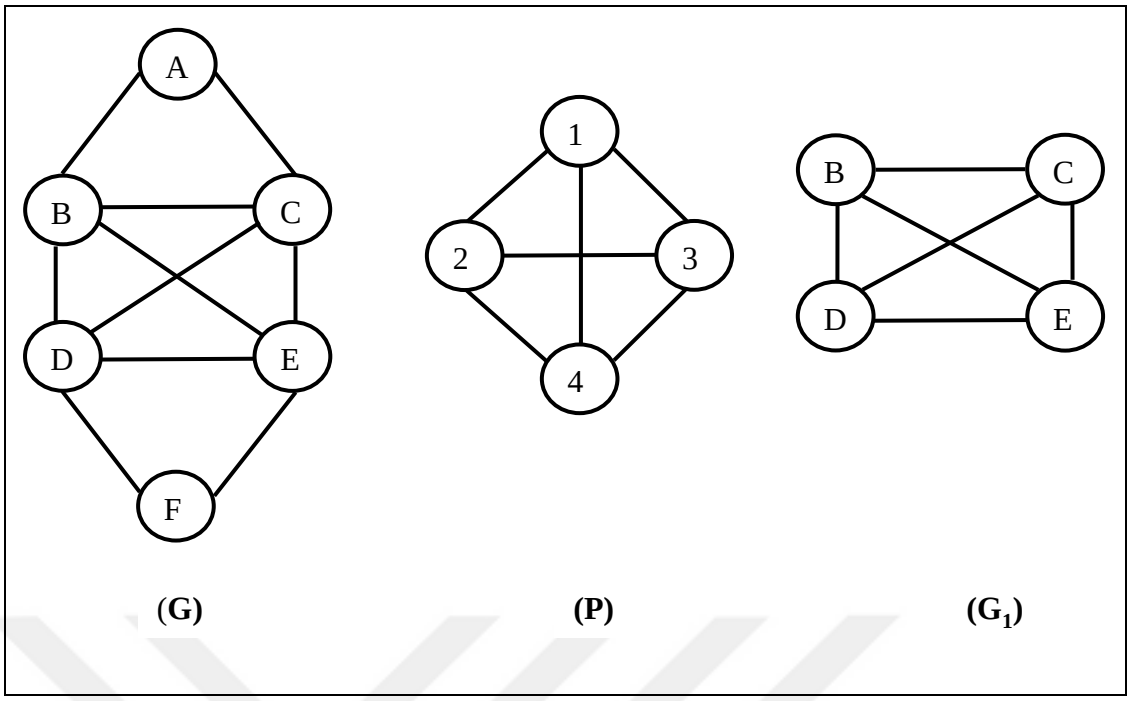
Patojen ile insan protein etkileşim ağları analiz edilirken veri kümesinin doğasına yakınlığı sebebi ile çizge modellemesinden sıklıkla yararlanılmaktadır. Bu çizgeler analiz edilirken araştırmacıların yıllar içerisinde geliştirdiği çizge analiz yöntemlerinden faydalanılmaktadır. Bu yöntemlerin bazıları çizgeleri bütüncül olarak ele alır ve bu çizgeyi ifade edebilecek tek bir anlam yüklemeye çalışır. Bir diğer yöntem kümesi ise çizgeleri, içerisinde barındırdığı tekrarlı veya seçkin alt bölümlerine göre incelemeyi hedefler. Birçok alanda olduğu gibi çizgeler üzerinde de alt yapılara göre yapılan analizlerin başında örüntü tanıma işlemi gelmektedir. Örüntü tanıma ile yapılan analizler, farklı patojenlerin insan proteinlerini benzer yapısal ilişkiler ile etkileyip etkilemediğini ortaya çıkarma imkânı sağlamaktadır.

Yaygın analiz yaklaşımlarından biri olan örüntü tanıma: Bilgisayar algoritmaları kullanarak bir veri kümesi içerisinde belirli bir örüntünün belirlenmesi ve daha sonra bu örüntü bilgisinin veri kümesinin sınıflandırılması gibi amaçlarla

kullanılması olarak tanımlanmaktadır [Bishop, 2006]. Örüntü tanıma yönteminin çizgelere uygulanması hem analiz tekniğinin hem de veri modelinin popülerliği sayesinde 70'lerin başlarına kadar dayanmaktadır. Araştırmacılar yıllardan beri kimyasal bileşiklerin atom yapıları [Willet et al., 1998], proteinler arası etkileşimler [Tian et al., 2006], sosyal ağlar [Cai et al., 2005], görüntü işleme [Liu and Lee, 2001] gibi önemli alanlara ait veri kümelerinin modellendiği çizgelerde örüntü tanıma problemine etkin çözüm yöntemleri geliştirmektedirler. Corneil [Corneil and Gotlieb, 1970] 1970 yılında örüntü tanıma probleminin çizgeler üzerinde uygulanması işleminin adını özelleştirerek alt-çizge eşbiçimlilik (İng. Subgraph Isomorphism) ismini vermiştir. Yine aynı makalede alt-çizge eşbiçimlilik problemi, "verilen G_1 ve G_2 çizgelerinden, G_1 çizgesinin, G_2 çizgesine ait bir alt çizgeye eşbiçimli olup olmadığının belirlenmesi" olarak tanımlanmıştır. Yıllar içerisinde problemin tanımı genişletilerek G_2 çizgesi içerisinde, G_1 ile eşbiçimli olan tüm alt-çizgelerin bulunması problemine evirilmiştir.

Problemin doğası gereği arama uzayı oldukça geniştir ve eşbiçimli altçizge olup olmadığının bulunması problemi SAT veya Hamilton çevrimi bulma gibi NP-Tam bir probleme indirgenebildiği için NP-Tam bir problem olarak kabul edilmektedir [Cook, 1971]. NP-Tam problemleri çözen doğrusal zamanlı bir algoritma henüz bilinmediği için alt-çizge eşbiçimlilik probleminin çözümünde yüksek verimli, sezgisel iyileştirmelere sahip algoritmalar kullanılmaktadır.

Geliştirilen algoritmalar çözüm ürettikleri problem türüne göre temelde iki ana sınıfa ayrılmıştır. Şekil 1.1'de gösterilen eşleme yöntemine birebir eşleme ismi verilmektedir. Bu problem türünde; algoritmalar Corneil'in genişletilmiş tanımına uygun olarak, hedef çizge içerisindeki sorgu çizgesi ile eşbiçimli olan tüm alt-çizgelerin bulunması hedeflenmektedir. Bu problem için geliştirilen çözüm yaklaşımları kullanılarak farklı patojenlerin sebep olduğu protein etkileşim ağlarında belirli bir etkileşim yapısının birebir benzer şekilde bulunup bulunmadığı veya bulunma sıklığı analiz edilebilmektedir.



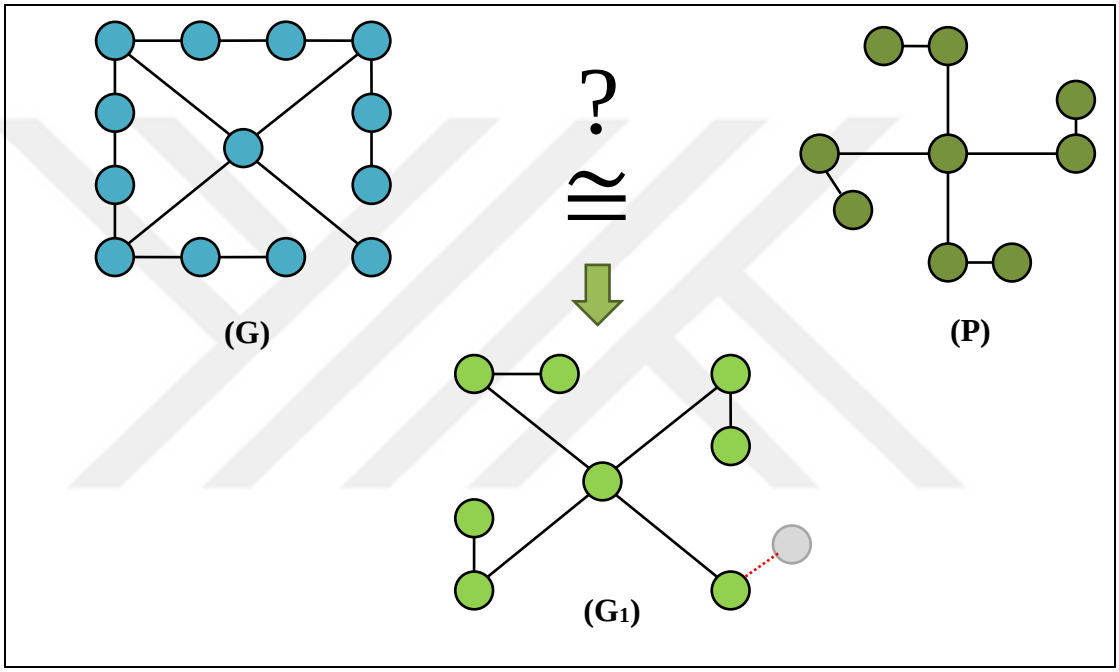
Şekil 1.1: *G* hedef çizgesi içerisinde, *P* sorgu çizgesi ile birebir eşbiçimli olan G_1 alt-çizgesi.

Alt-çizge eşbiçimlilik problemi için alanda öncü kabul edilen Ullmann'ın Algoritması [Ullmann J. R., 1976] da dahil olmak üzere birçok birebir eşleme algoritması geliştirilmiştir. Bu algoritmalar aynı işlemi yerine getirse de kullandıkları yöntemler farklılık gösterebilmektedir. Birebir eşleme algoritmaları, yaklaşımlarına göre: dal-ve-sınır algoritmaları ve indeks tabanlı algoritmalar olarak ikiye ayrılmaktadır. Dal-ve-sınır yaklaşımına sahip algoritmaların hafıza gereksinimi düşüktür ve küçük-orta ölçekli çizgelerde etkilidirler. İndeks tabanlı algoritmalar ise hatırı sayılır miktarda hafıza kullanmasının yanında, yüksek çalışma süresi performansları ile büyük çizgelerde alt-çizge eşbiçimlilik problemini çözmeyi mümkün kılmaktadır. Bu çeşitlilik sayesinde farklı gereksinimleri olan sistemlerde birebir eşleme işlemini etkili şekilde gerçekleştirme imkânı doğmaktadır.

Güvenilir ve eksiksiz sonuçlar üretmesi açısından oldukça popüler olan birebir eşleme yaklaşımı, veri kümesinin güvensiz veya eksik olduğu durumlarda doğası gereği yeterli olmamaktadır. Örneğin, patojen – insan protein etkileşim ağları oluşturulurken eksik gözlemlenen protein etkileşimleri olabilmektedir. Bu durumda birebir eşleme probleminin bu ağlar üzerinde uygulanması eksik veya hatalı analiz yapmaya sebebiyet verecektir. Bu tür çizgeler üzerinde de anlamlı alt-çizge eşbiçimlilik sorguları yapılabilmesi adına probleminin tanımı esnetilerek probleme

yeni bir yaklaşım getirilmektedir. Yaklaşık alt-çizge eşbiçimlilik olarak adlandırılan bu problem türünde, hedef çizgenin sorgu çizgesi ile kabul edilebilir benzerlikteki alt-çizgeleri aranmaktadır.

Şekil 1.2 de örnek bir G hedef çizgesi ve P sorgu çizgesi gösterilmektedir. G hedef çizgesi içerisinde P sorgu çizgesi ile birebir eşbiçimli alt-çizge sorgusu yapıldığında herhangi bir sonuç bulunamayacaktır. Diğer taraftan, eğer birebir eşleme işlemi bir ayrıt ve bir düğüm eksikliğine tahammüllü hale getirilirse G_1 alt-çizgesi bu eşleme işlemine uygun bir sonuç olarak bulunabilecektir.



Şekil 1.2: *G* hedef çizgesi içerisinde, *P* sorgu çizgesi ile yaklaşık eşbiçimli olan G_1 alt-çizgesi.

Esnek eşlemeyi mümkün kılabilmek için algoritmalar arama uzaylarını birebir eşleme yaklaşımına göre oldukça genişletmelidirler. Genişleyen arama uzayı ile başa çıkmak için ise yüksek işlem gücüne ihtiyaç duymaktadır. 2000'li yıllardan sonra artan işlem güçleri sayesinde bu yaklaşımın popülerliği artmıştır [Nilsson, 1980]. Arama uzayının bu denli büyük olması, çok büyük çizgelerde yaklaşık eşlemeyi imkânsız hale getirmektedir. Bu sebeple büyük çizgelerde yaklaşık eşleme probleminin çözümü için farklı bir bakış açısı geliştirilmiştir. Bu yeni yaklaşıma sahip algoritmalar belirli benzerlikteki tüm alt çizgeleri bulmak yerine, sorgu alt-çizge ile en çok benzerlik gösteren *k* adet alt-çizgeyi bulmayı hedefler. Bu yaklaşım sayesinde büyük çizgelerde yaklaşık eşleme yapılabilmektedir çünkü yöntem tüm

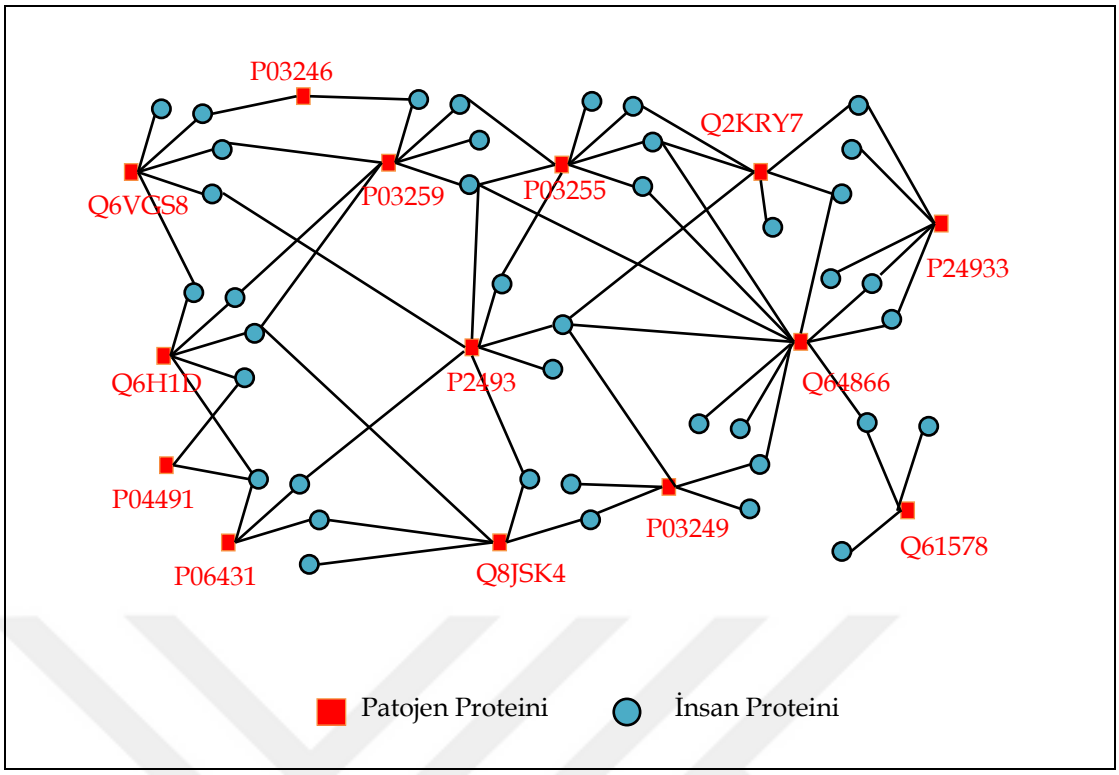
uzayı aramak yerine sadece uzayda kendisine yakın k adet komşuyu bulmayı hedeflemektedir.

Şimdiye kadar bahsedilen problemler ve bu problemlerin için geliştirilmiş yaklaşımlar zaman içerisinde ortaya çıkan ihtiyaçlar doğrultusunda geliştirilmiştir ve geliştirilmeye devam etmektedir. Günümüzde çizge boyutları milyonları aşmaktadır [Cordella et al., 2004] ve bu sebeple alt-çizge eşbiçimlilik probleminin bu büyüklükte çizgeler üzerinde çözülmesi gittikçe zorlaşmaktadır. Araştırmacılar artan çizge boyutları ile başa çıkabilmek için çizgelerin yapısal, anlamsal ve karakteristik özelliklerinin tümünden fayda sağlamaya çalışmaktadır. İkinci bölümde detaylıca bahsedilecek olan algoritmalar, eşleme işlemini hızlandırma amacıyla; çizgelerin düğüm dereceleri veya düğüm komşulukları gibi yapısal özelliklerinden, düğüm veya ayrıt etiketi gibi anlamsal özelliklerinden ya da ayrıtlarının yönlü olması veya düğümlerinin ağırlıklı olması gibi çizge karakteristiklerinden faydalanmaktadır.

1.1. Tezin Amacı, Katkısı ve İçeriği

Enfeksiyon hastalıklarına sebep olan patojen proteinleri ile insan proteinleri arasındaki etkileşimleri içeren veri kümesi çizge olarak modellendiğinde bu çizgenin iki parçalı çizge (İng. Bipartite Graph) karakteristiğine sahip olduğu görülmektedir. Bu tip çizgeler üzerinde alt-çizge eşbiçimlilik probleminin çözümü için literatür taraması yapıldığında, bildiğimiz kadarıyla, iki parçalı çizgeler için özelleşmiş bir çözüm yaklaşımına rastlanmamaktadır. Şekil 1.3'de görülen protein etkileşim ağları, seçim verileri, biyomedikal veriler, kötücül yazılım tespit veri kümeleri, bulut hesaplama ağları vb. [Stauffer et al., 2017] birçok önemli veri kümesi iki parçalı çizge olarak modellenmektedir. Bu denli önemli alanlarda kullanılması ve problemin iki parçalı çizgeler üzerinde de NP-Tam [Kijima et al., 2012] bir problem olması sebebiyle problemin çözünün hızlandırılması için pratikte uygulanabilecek geliştirmeler oldukça değerli olacaktır.

Önceden bahsedildiği üzere; artan çizge boyutları sebebi ile çizgelerin her türlü özelliği kullanılarak alt-çizge eşleme performansı artırılmaya çalışılmaktadır. Araştırmacılar bu amaçla sadece özel bir çizge türünde yüksek performans ile alt-çizge eşbiçimlilik problemini çözebilen algoritmalar geliştirmişlerdir.



Şekil 1.3: Bir patojene ait proteinler ile insan proteinleri arasındaki etkileşim ağı örneği.

2008 yılında yapılan bir çalışmada sadece yönlü çizgeler üzerinde çalışabilen bir yöntem geliştirilmiştir [Cheng et al., 2008]. Geliştirilen bu algoritma ile genel çizgeler için geliştirilen algoritmalar kıyaslandığında; performans bakımından çizge için özelleşmiş yöntemin daha verimli olduğu görülmektedir. Bu motivasyonla iki parçalı çizgeler için özelleşmiş, bu çizgelerin yapısal özelliklerini hesaba katan bir algoritmanın genel çözüm yaklaşımlarından daha hızlı olacağı öngörülmektedir.

İki parçalı çizgeler üzerine yazılan kapsamlı kitapta [Asratian et al., 1998] bahsedildiği üzere: Bu tür çizgelerin ikili yapısı kullanılarak birçok problemin arama uzayı daraltılabilmektedir. Bu problemlerden biri de yapılan tez çalışmasını oldukça yakından ilgilendiren çizge eşbiçimlilik problemidir. Hopcroft'un [Hopcroft and Karp, 1973] yayımladığı makalesi bu görüşü kanıtlar niteliktedir. Hopcroft, iki parçalı çizgelerin birebir eşlenmesinin $O(n^{5/2})$ zaman karmaşıklığında mümkün olduğunu göstermektedir. Diğer yandan, bazı yöntemler yaklaşık eşleme yaparken, çizgeler arasındaki benzerliğin ölçütü olarak maksimum iki parçalı alt-çizgeleri kullanmaktadırlar. Bu çalışmalardan biri olan, Fankhauser ve arkadaşlarının yayımladığı makalede [Fankhauser et al., 2012] normal çizgeler, eşbiçimlilik sorgusu

daha az maliyetli olduğu için: Çizgeleri öncelikle maksimum iki parçalı alt-çizgelere dönüştürmekte daha sonra eşlenme yapmaktadır.

Tüm bu motivasyonlara dayanarak iki parçalı çizgeler üzerinde birebir ve yaklaşık eşleme problemlerinin incelenmesinin çalışmaya değer bir alan olduğuna kanaat getirilmiştir. Geliştirilecek algoritmalar sadece bu tip çizgelerde çalışabiliyor olsa da iki parçalı çizgelerin kendine has yapısı kullanılarak problemin çözüm performansının oldukça arttırabileceği öngörülmektedir.

Giriş bölümünde; alt-çizge eşbiçimlilik problemine dair giriş bilgisi, tez çalışmasının amacı ve katkılarından bahsedilmektedir. İkinci bölümde; giriş bölümünde değinilen problem, tüm alt problemleri ve bu problemlere geliştirilen çözümlerle beraber detaylı biçimde anlatılacaktır. Problemler anlatılırken, bu problemlerin çözümüne dair literatür çalışmalarının benzerliklerine ve farklılıklarına değinilecektir. Üçüncü bölümde tez çalışması kapsamında iki parçalı çizgeler için geliştirilen birebir eşleme yaklaşımları açıklanmaktadır. Dördüncü bölümde ise yine yapılan çalışmada iki parçalı çizgeler için geliştirilen yaklaşık eşleme algoritması anlatılmaktadır. Beşinci bölümde; bu çalışma kapsamında geliştirilen algoritmaların, literatürdeki genel çizgeler için geliştirilmiş yüksek performanslı emsalleri ile performans karşılaştırmalarına yer verilmektedir. Son olarak altıncı bölümde tez çalışmasından çıkarılan sonuçların özeti bu tez çalışmasını referans alacak araştırmacılara gelecek çalışmalar üzerine tavsiyeler bulunmaktadır.

2. PROBLEM TANIMI VE İLGİLİ ÇALIŞMALAR

Çizgeler (Tanım 1), nesneler ve bu nesneler arasındaki ilişkileri içeren veri kümelerini modellemek ve görselleştirmek için kullanılan soyut veri yapılarıdır. Çizgelere ait bu görselleştirme özelliği, bu yapıları araştırmacılar tarafından oldukça popüler hale getirmektedir çünkü görselleştirilen veri kümeleri, insan beyni tarafından çok daha kolay anlamlandırılabilmektedir.

Tanım 1 (Çizge): Çizge, G = (V, E), sonlu iki kümeden oluşan matematiksel bir yapıdır: V ile gösterilen küme çizgeyi oluşturan öğeleri (düğümleri), v, içeren küme iken; $E \subseteq V \times V$ kümesi bu öğeler arasındaki ilişkileri belirten ayrıtları, e, barındıran kümedir.

Araştırmacılar, bu veri yapılarını analiz etmek, sınıflandırmak, anlamlandırmak vb. işlemler için birçok yöntem geliştirmiştir. Bu yöntemlerin bazıları çizgeleri bir bütün olarak ele alır ve üzerinden bilgi çıkarımı yapmayı hedefler. Çizge eş-biçimlilik problemi (Tanım 2) olarak bilinen problem, bahsi geçen yöntemlerin hedefinde olan popüler çizge problemlerinden biridir. Bu problem, iki veri çizgenin birbirleri ile nitelik ve nicelik bakımından aynı olup olmadığını belirlemekte kullanılmaktadır.

Tanım 2 (Çizge Eşbiçimlilik): G = (V, E) ve G' = (V', E') olmak üzere iki çizge verilmiş olsun. G çizgesindeki her bir düğümü G' çizgesinde bir düğüme haritalayabilecek birebir ve örten bir fonksiyon $I: V \rightarrow V'$ varsa ve her u, $v \in V$ için $(u, v) \in E$ ancak ve ancak $(I(u), I(v)) \in E'$ ise G ve G' çizgeleri birbirleri ile eşbiçimlidir. G ile G' çizgelerinin eşbiçimlilik durumu $G \cong G'$ ifadesi ile gösterilir.

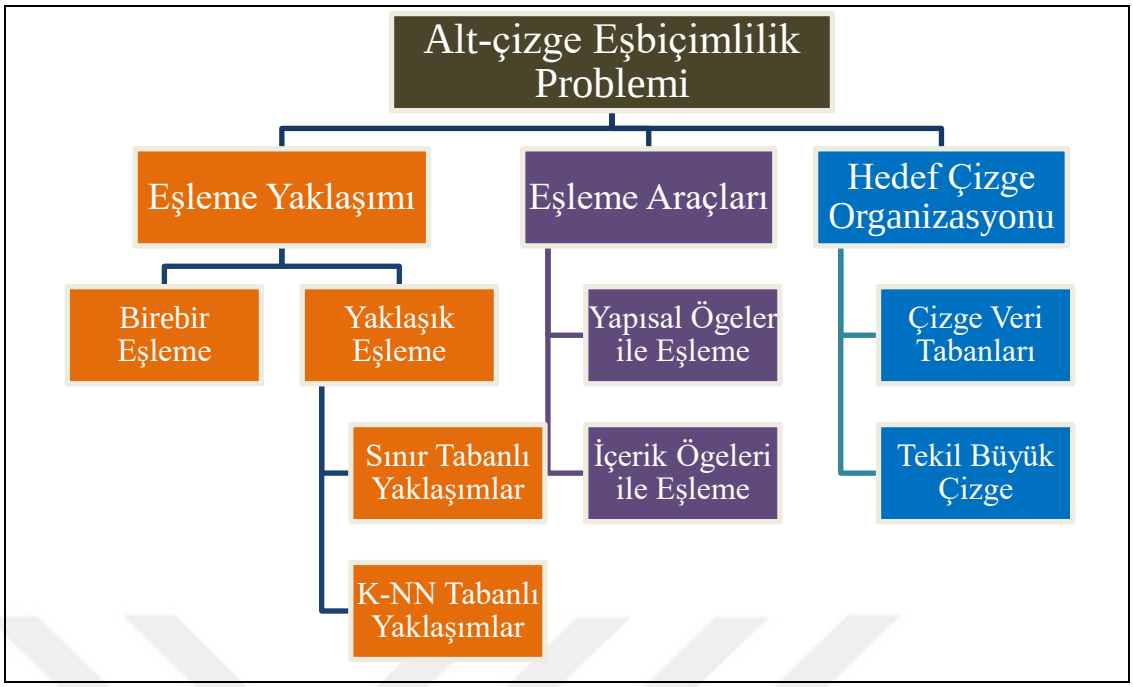
Çizgeleri genel tek bir yapı olarak ele alan yaklaşımların aksine bazı yöntemler, çizgeleri birçok altyapıdan oluşan birleşik bir yapı olarak ele almaktadır. Çizge içerisindeki bahsi geçen bu alt yapılara alt-çizge ismi verilmektedir:

Tanım 3 (Alt-çizge): Alt-çizge $H = (V', E')$; bir çizgenin $G = (V, E)$ içerisinde bulunan ve tüm düğümleri G çizgesinin de düğümü, $V' \subseteq V$ ve tüm ayrıtları, $E' \subseteq E \cap (V' \times V')$ olacak şekilde G çizgesinin de ayrıtı olan çizgelerdir.

Çizge analizlerini, alt-çizgeler üzerinden yapmak oldukça yaygın kullanılan bir yaklaşımdır. Çizge içerisinde sık beliren alt-çizgelerin bulunması ya da belirli alt-çizge yapılarının çizge içerisindeki yoğunluğunun araştırılması gibi problemlerin çözümüne yönelik literatürde birçok çalışma bulunmaktadır. Bahsi geçen problemlerle benzer bir diğer popüler problem ise alt-çizge eşbiçimlilik problemidir:

Tanım 4 (Alt-çizge Eşbiçimlilik Problemi): $P = (V_1, E_1)$ ve $G = (V_2, E_2)$ olmak üzere, $|V_1| \leq |V_2|$, $|E_1| \leq |E_2|$ koşulunu sağlayan iki çizge verilmiş olsun. $P \cong H$ koşulunu sağlayan, G'nin tüm H alt-çizgelerinin bulunması problemine alt-çizge eşbiçimlilik problemi denir.

Bu problemlerin çözümüne yönelik çalışmalar, kırk yılı aşkın bir süredir, dünyanın her yerinden araştırmacılar tarafından özenle yürütülmektedir. Corneil'in, hedef çizge içerisinde sorgu ile eşbiçimli bir çizgenin bulunup bulunmadığının belirlenmesi olarak tanımladığı problem; yıllar içerisinde genişletilerek eşbiçimli olan tüm alt-çizgelerin bulunması problemine dönüşmüştür.



Şekil 2.1: Alt-çizge eşbiçimlilik problem türleri diyagramı.

Problem yıllar içerisinde birçok varyasyona ayrılmıştır [Yan et al., 2004]. Şekil 2.1'de alt-çizge eşbiçimlilik probleminin çeşitli ögelere göre dallandığı alt-problem türleri gösterilmektedir. Bu varyasyonlar eşleme işleminin hangi araçlarla veya nasıl yapılacağına ya da eşleme için hedef alınan veri kümesinin çizge organizasyonuna göre oluşmaktadır. Varyasyonlar da yine kendi içlerinde kullandıkları algoritma tiplerine göre alt kümelere ayrılabilmektedir. Tezin bu bölümünde; varyasyonların ve bunlara ait alt çözüm kümelerinin ne şekilde oluştuğu, farklılıkları ve ortak yönleri anlatılmaktadır. Her bir alt başlık, literatürdeki çalışmalardan örnekler verilerek zenginleştirilecektir.

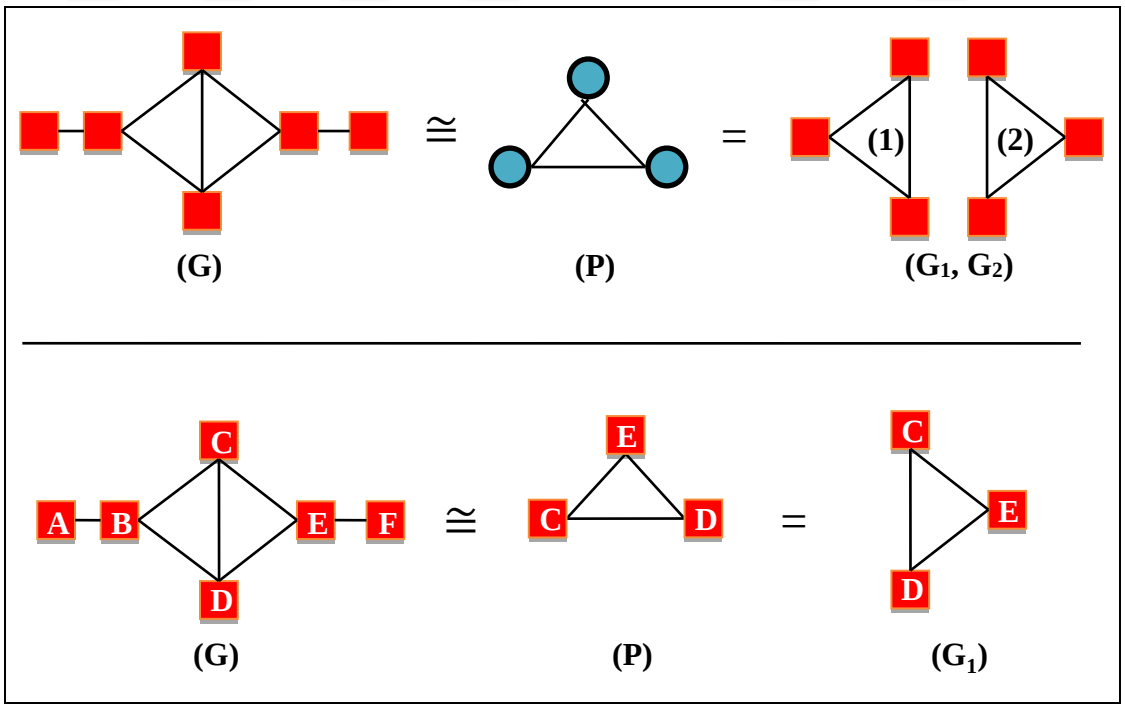
2.1. Yapısal veya İçerik Ögeleri ile Eşleme

Çizgeler iki temel öge kümesinden oluşmaktadır: Düğüm, ayrıt, komşuluk gibi yapısal ögeler ve düğüm/ayrıt etiketi, ağırlıkları gibi içerik ögeleri. Algoritmalar eşleme yaparken ihtiyaçları doğrultusunda içerik ögelerini dikkate alır ya da daha genel bir çözüm için bu ögeleri göz ardı edebilir. Yapısal eşleme yapan bir algoritma, eşleme sırasında sadece çizgeyi oluşturan düğüm ve ayrıtların sayılarının eşit olmasına ve komşulukların (Tanım 5) aynı olmasına dikkat eder. İçerik ögeleri ile eşleme yapan algoritmalar, çizgenin yapısal özelliklerinin yanı sıra yapısal

elemanlara ait nitelikleri de eşlemeye çalışmaktadır. Örneğin; bir sorgu çizgesi düğümünün: etiketi, ağırlığı ve eğer varsa düğüm tipi bilgisi gibi niteliklerinin, haritalandığı hedef çizge düğümünde de aynı olması gerekmektedir.

Tanım 5 (Düğüm Komşuluğu): Her bir ayrıtın e = (u, u') ∈ E birbirine bağladığı bir çift düğüme komşu düğüm denir. Bir u düğümünün komşu düğümlerinin kümesi N(u) = {u' ∈ V | ∃ (u, u') ∈ E}, o düğümün komşuluk kümesi olarak adlandırılır.

Problemin çözümü için geliştirilen ilk algoritma olan Ullmann'ın Algoritması, çizgeleri eşlerken bir düğümden başlayarak uygun düğüm derecesine sahip tüm hedef düğümler üzerine dallanmaktadır. Uygun uzunlukta dallanmaların komşulukları geri izleme ile kontrol edilerek uygun sonuçlar belirlenmektedir. Görüldüğü üzere Ullmann sadece düğümler ve bu düğümler arasındaki komşulukları kullanarak yapısal bir eşleme yapmaktadır. Birçok algoritma [Yan and Yu, 2005], [Solnon, 2010], [Giugno and Shasha, 2002], [Jiang et al., 2007], [Shang et al., 2008] hedef çizge ile sorgu çizgesi arasında anlamsal olarak hiçbir bağlantı olmadığı durumlarda da yapısal olarak bir eşleme çözümü sunmaktadır.



Şekil 2.2: G hedef çizgesi ile P sorgu çizgesinin yapısal (üstte) ve içerik (altta) ögelerine göre eşlenmesinden elde edilen sonuçlar.

Yapısal eşleme yapan algoritmalar yaklaşımın doğası gereği nitelikli çizgeler (Tanım 6) üzerinde de probleme çözüm üretmektedir. Bu tür algoritmalar içerik tabanlı eşleme yapan algoritmalardan [Zou et al., 2008], [Zhang et al., 2009], [He and Singh, 2008], [Zhao and Han, 2010], [Han and Lee, 2013] daha fazla sonuç üretmektedir çünkü içerik tabanlı eşleme yöntemlerin aynı zamanda yapısal eşleme kıstaslarına da biat etmesi gerekmektedir. Bu sebeple, yapısal eşleme probleminin çözümü içerik ögelerine göre eşleme probleminin çözümünden daha maliyetlidir.

Şekil 2.2'de görüldüğü üzere; yapısal eşleme yapıldığında iki farklı, G_1 ve G_2, eşbiçimli alt-çizge bulunmaktadır. İçerik ögeleri dikkate alındığında, düğüm etiketleri sebebiyle sadece bir eşbiçimli alt-çizge bulunabilmektedir. Bu avantaj aynı zamanda algoritmaların çalışma zamanları göz önüne alındığında bir dezavantaja dönüşmektedir. Yapısal eşleme yapan algoritmalar çizgelerin içerik özelliklerinden faydalanamadığı için arama uzayı diğer yaklaşıma göre oldukça geniştir ve bu sebeple daha uzun çalışma sürelerine ihtiyaç duyarlar.

Tanım 6 (Nitelikli Çizgeler): Ayrıt ve düğümleri anlamsal bilgiler (etiket, ağırlık vb.) taşıyan çizgelerdir. Bir nitelikli çizge $G = (V, E, \beta, \theta)$, düğüm kümesi V, ayrıt kümesi E, düğüm nitelik fonksiyonu β ve ayrıt nitelik fonksiyonu θ'dan oluşmaktadır. Bir çizgede sadece düğüm etiketleri nitelik olarak bulunuyorsa bu çizge $G = (V, E, L)$ olarak gösterilmektedir ve L, her bir düğüme ait etiket, l, değerini bildiren düğüm etiket fonksiyonudur.

Alt-çizge eşbiçimlilik araması yapılmak istenen problem iyi analiz edilmeli ve ihtiyacı en iyi karşılayan yöntem kullanılmalıdır. Örneğin; sosyal-ağ analizinde kişiler arası ilişkiler incelenirken birbiri ile sadece bir ortak arkadaşı olan kişileri aramak için yapısal eşleme yapan bir algoritmaya ihtiyaç duyulmaktadır. Diğer yandan, karmaşık bir kimyasal bileşik ağında özel bir atom ilişki yapısının vuku bulduğu alanlar araştırılıyorsa içerik tabanlı eşleme yapmak gerekmektedir. Bahsi geçen avantaj ve dezavantajları göz önünde bulundurulduğunda; ihtiyaca göre her iki yaklaşımın da yaygın olarak kullanıldığını söylemek mümkündür.

2.2. Çizge Veri tabanları – Tek Büyük Çizge ile Eşleme

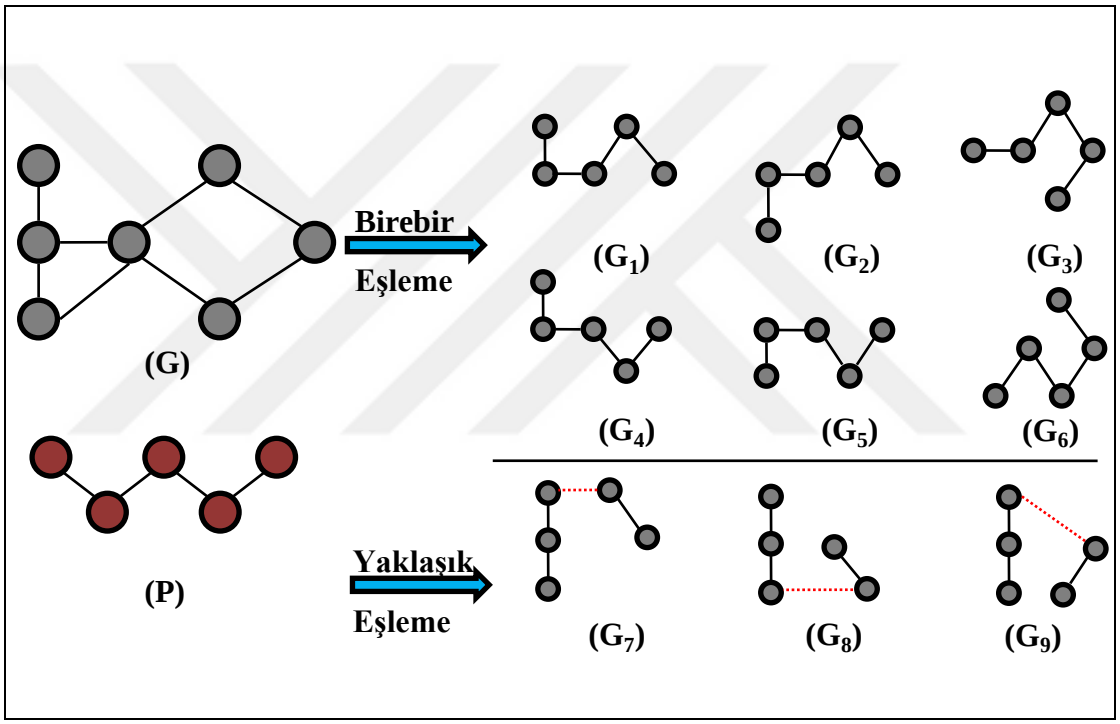
Alt-çizge eşbiçimlilik araması yapılmak istenen hedef yapı, sosyal-ağlar gibi içerisinde milyonlarca düğüm bulunduran tek bir çizge veya protein etkileşim ağları gibi yüzlerce düğüm içeren birçok çizgeden meydana gelen bir çizge veri kümesi olabilmektedir. Tek, büyük bir çizgeyi hedef alacak şekilde tasarlanan algoritmalar [Ren and Wang, 2015], [Bi et al., 2016], [Cheng et al., 2007], [Chen, ve diğerleri, 2007] [Xie and Yu, 2011] genellikle çizgeler üzerinde bir alan tarama işlemi gerçekleştirerek eşleme yapabileceği alt alanları belirler. Algoritmalar daha sonra bu alt alanlar içerisinde eşbiçimli alt-çizgeleri bulmaktadır. Çizge veri tabanlarına yönelik geliştirilen yöntemlerde [Lin et al., 2011], [Yuan et al., 2012], [Ullmann R. , 2010], [Bonnici et al., 2013], [Čibej and Mihelič, 2015] ise genellikle çizgeler bir şekilde önişlemden geçirilerek o çizgeyi temsil edebilen bir tür bilgi ile etiketlenir. Sorgu çizgesinden elde edilen bilgi ile bu çizge veri kümesi filtrelenir ve kalan aday çözümler içerisinde alt-çizge eşbiçimlilik araması yapılır. Her iki yöntemde sıklıkla kullanılmaktadır. Örneğin; insan proteinleri ile patojen proteinleri arasındaki etkileşim ağları, Şekil 1.3'deki gibi binlerce küçük ve orta ölçekli çizgeden oluşmaktadır. Diğer yandan insana ait proteinlerin insan metabolizmasında kendi aralarında yaptıkları protein etkileşimlerinin ağı tek bir büyük çizge ile modellenmektedir.

2.3. Birebir Eşleme – Yaklaşık Eşleme

Çizgeler üzerinde alt-çizge eşleme işlemi yapılırken, algoritmaların eşleme için belirlediği yeterlilik koşullarına göre bulduğu sonuçlar değişebilmektedir. Yapısal eşleme yaklaşımı ile birebir eşleme yapmak için: Sorgu çizgesi ile hedef çizgeye ait bir alt-çizge arasında; düğümler ve bu düğümlere ait komşuluklar açısından eksiksiz bir haritalama olmalıdır (Tanım 3). Diğer yandan eğer içerik ile birebir eşleme yapılıyorsa; eşleme için yapısal eşleme kıstaslarına ek olarak dikkate alınan içerik özelliklerinin tamamının (düğüm ağırlıkları, ayrıt ağırlıkları, düğüm etiketleri, ayrıt yönleri vs.) da birebir örtüşmesi gerekmektedir.

Bazı durumlarda birebir eşleme ihtiyaçları tam olarak karşılayamaz. Örneğin içerisinde gürültü bulunan bir hedef alt-çizge eşbiçimlilik araması yaparken belirli

bir oranda hataya tolerans gösterilmesi gerekmektedir, aksi halde istenilen sonuçlar çizgede bulunsa dahi birebir eşleme algoritması bunları bulamayacaktır. Diğer yandan birebir eşleme algoritmaları, tüm sonuçları eksiksiz bulmak için geliştirilmiştir fakat çok büyük çizgelerde birebir eşleme yapmak oldukça fazla kaynak ve zaman gerektirmektedir. Bu tür durumlarda sorgu çizgesi ile en eşbiçimli belirli sayıda sonucun hızlı bir şekilde üretilmesi, problemi bu tür çizgelerde çözülebilir hale getirmektedir. Bahsi geçen durumlar ile başa çıkmak için araştırmacılar, literatüre yaklaşık eşleme yöntemleri adıyla geçen yöntemler geliştirmişlerdir.



Şekil 2.3: G hedef çizgesi ile P sorgu çizgesinin birebir eşleme (sağ üst) ve bir ayrıt tahammüllü yaklaşık eşleme ile üretilen ek sonuçlar (sağ alt).

Bu yöntemlerde bir eşlemeyi geçerli kabul etmek için sorgu çizgesi ile hedef alt-çizge arasında birebir eşleşme şartlarının eksiksiz sağlanması beklenmez. Eşleşme sırasında, eşlenmesi beklenen öğelerin eksikliğine, fazlalığına veya yanlış eşleşmesine izin verilerek eşleme işlemi esnetilmektedir. Şekil 2.3'te hedef çizge içerisindeki G_1 – G_6 alt-çizgeleri sorgu çizgesi ile birebir eşlenebilmektedir. Diğer yandan $G_{7\text{-}9}$ alt-çizgeleri ile de eşleme yapabilmek için bir ayrıtın eksikliğine tahammül gösterilmesi gerekmektedir. G hedef çizgesi bir protein etkileşim ağının modeli ise; eksik ayrıt, deneyleri gerçekleştiren bilim insanın gözünden kaçan bir

ilişki olabilir. Bu durumda yaklaşık eşleme yapıldığında Şekil 2.3'de görüldüğü birebir eşbiçimli olan alt-çizgelere ek olarak $G_{7\text{-}9}$ alt-çizgeleri de sonuçlar arasında yer alacaktır.

Probleme ait bu varyasyon çeşitlenmesi problemin çözümü için geliştirilmiş algoritmaları sınıflandırırken en yaygın çeşitlenmedir ve bu sebeple algoritmalar öncelikle birebir veya yaklaşık eşleme algoritması olarak anılmaktadır. Her iki problem yaklaşımı için geliştirilen yöntemler, probleme yaklaşım şekline göre alt kümelere ayrılmaktadır. Bir sonraki bölümde problem yaklaşımlarına alt başlıklar halinde değinilecektir. Problem yaklaşımlarına geliştirilen çözüm yöntemleri ortak yönleri ve farklılıkları ile detaylı olarak anlatılacaktır.

2.3.1. Birebir Eşleme Yaklaşımı

Alt-çizge eşbiçimlilik problemine geliştirilen öncü çözüm yaklaşımıdır. Bu yaklaşım genellikle çizgelerin doğruluğundan ve tamlığından emin olunan durumlarda kullanılmaktadır çünkü yaklaşımın amacı: Sorgu çizgesinin tüm yapısal ögelerine (düğüm, ayrıt vb.) birebir ve aynı yapıdaki hedef çizgeye ait alt-çizgeleri bulmaktır (Tanım 4). Uygun tüm alt-çizgelerin bulunması zorlu bir işlemdir çünkü arama uzayı çizge boyutuyla üstel orantılı olarak artmaktadır. Geliştirilen öncü algoritmaların başa çıkmaya çalıştığı çizge boyutları küçük olduğu için arama uzayının büyümesi çok büyük bir problem olarak görülmemiştir. Bu sebeple öncü algoritmalar genellikle açgözlü budama tekniklerine sahip, arama uzayı geniş dal-ve-sınır algoritma yaklaşımı göstermektedir. Artan çizge boyutları, problemin bu algoritmalar ile çözümünü zorlaştırmakta hatta imkânsız hale getirmektedir. Araştırmacılar artan çizge boyutları ile başa çıkabilmek için daha verimli algoritmalar oluşturmaktadır. Bu algoritmalar öncü algoritmalar gibi açgözlü kararlarla değil çizgenin bütününü ele alan verimli yöntemler ile arama uzayını küçülterek çözüm üretmektedir. Birebir eşleme yapmak için geliştirilen algoritmalar temelde bu iki yaklaşım tipine göre alt kümelere ayrılmaktadır.

Dal ve Sınır yaklaşımı, problemin çözümüne yönelik ilk algoritma olan Ullmann'ın Algoritmasının da sahip olduğu; hafıza tüketimi düşük fakat arama uzayı geniş bir çözüm tekniğidir. Bu çözüm tekniğine sahip algoritmalar [Carletti et al., 2015], [Asiler and Yazıcı, 2017], [Carletti et al., 2017], [Di Natale et al., 2010],

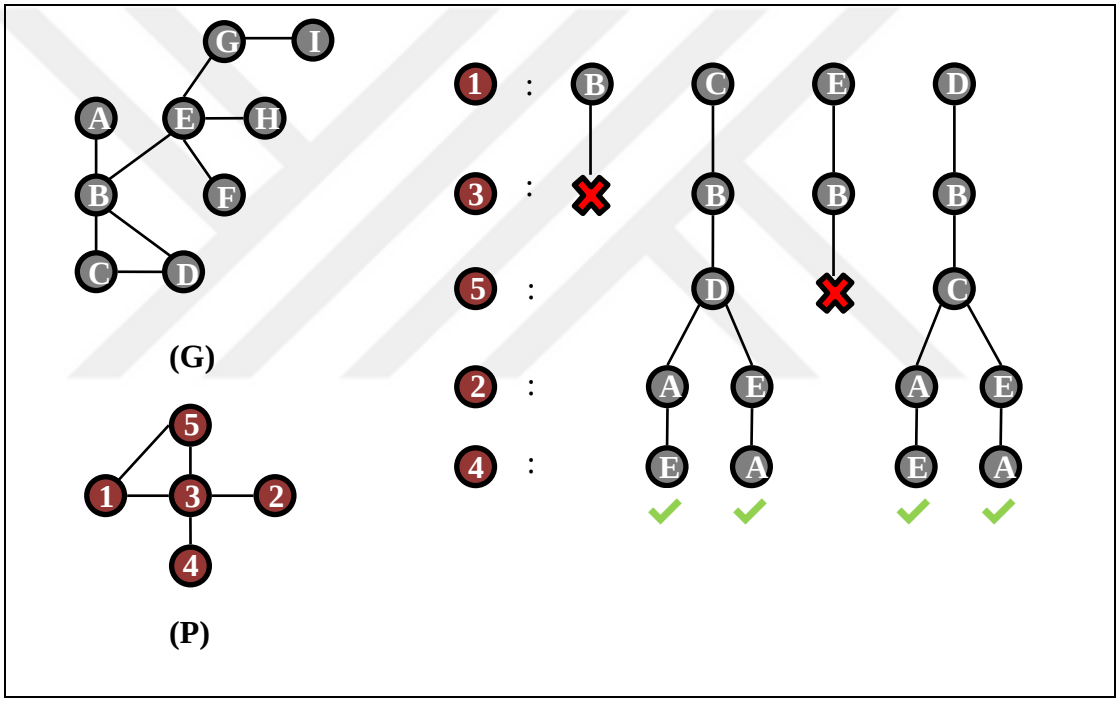
[Giugno et al., 2013], [Messmer and Bunke, 1995] sorgu çizgesindeki bir düğümü, hedef çizgedeki bir aday düğüm ile eşleyerek çözüme başlar. Her bir adımda mevcut çözüme yeni bir düğüm ekleyerek çözüm dallandırılır ve bu dallanma işlemi sorgu çizgesinin tüm düğümleri için haritalama işlemi yapılana kadar devam eder. Her bir dallanmanın sorgu alt-çizgesi ile eşbiçimli olup olmadığı kontrol edilerek birebir eşleme sonuçları elde edilir. Böyle bir durumda herhangi bir budama işlemi yapılmaz ise; n düğüme sahip bir hedef çizge üzerinde m boyutlu bir alt-çizgenin eşbiçimli alt-çizgeleri aranırken, $C\binom{n}{m}$ adet dallanmanın oluşturulması gerekir. Her bir dallanmanın sorgu çizgesi ile eşbiçimlilik kontrolünün yapılması gerekmektedir. Açıkça görülmektedir ki düğüm sayısı yüzler seviyesinde iken bile böyle bir arama yaklaşım oldukça maliyetlidir.

Geliştirilen algoritmalar arama uzaylarını küçültmek ve sonuç üretmesi mümkün olmayan dallanmaların önüne geçmek için çeşitli iyileştirmeler yapmışlardır. Arama uzaylarını küçültmek için geliştirilen yöntemlerden biri iyi bir başlangıç düğümü seçmektir çünkü iyi bir başlangıç düğümü oluşabilecek birçok gereksiz dallanmanın önüne geçmektedir. Diğer bir geliştirme ise sonuç üretmesi imkânsız dallanmaların olabildiğince erken budanmasıdır. Faydasız dallanmalar ne kadar erken budanırsa; sonuç üretmesi imkânsız alt dallanmalar o kadar azalacaktır ve ayrıca bu dallanmalar üzerinde alt-çizge eşbiçimlilik kontrolü gibi ağır bir yük de ortadan kalkacaktır.

Araştırmacılar, iyi bir başlangıç düğümü seçimi yapmak ya da uygunsuz dallanmaların erkenden budanmasını sağlamak amacıyla çizgelerin yapısal ve içerik özelliklerinden faydalanmaktadırlar. Optimum başlangıç düğümü ve dallanma sıralamasını belirlemek için RI [Bonnici et al., 2013] algoritması; Dallanma sırasını belirlerken düğümlerin sorgu çizgesi içerisindeki merkezi olma durumunu ve düğüm etiket frekansını dikkate almaktadır. Ullmann gereksiz dallanmaları önlemek için, dallanmaları yaparken seçtiği adaylarda düğüm derecelerinden (Tanım 7) faydalanırken, Cordella [Cordella et al., 2004] bu faydayı biraz daha geliştirmiştir ve iki uzaklıklı komşulukları ve bunları derecelerini de hesaba katarak budama işlemini gerçekleştirmektedir. Daha sonra çıkan LAD [Solnon, 2010] ve VF3 [Carletti et al., 2017] algoritmaları bir ön işlem yapar. Bu algoritmalar, her bir düğüm için uygun olabilecek aday kümelerini, düğüm dereceleri, düğüm etiketleri ve komşulukları dikkate alacak şekilde seçer ve dallanma başlar.

Tanım 7 (Düğüm Derecesi): Bir u düğümünün komşuluk kümesi N(u) olsun; bu kümenin eleman sayısı |N(u)| o düğümün derecesini göstermektedir. Bir başka deyişle, bir düğüme bağlı olan ayrıtların sayısı o düğümün derecesini belirtmektedir. D(u) ile gösterilir.

Şekil 2.4'te gösterilen örnekte, en verimli dal-ve-sınır algoritmalarından biri olan VF2 algoritmasının dallanma adımları gösterilmektedir. VF2 algoritması başlangıç düğümü seçimi için herhangi yöntem önermediği için rasgele bir düğümden dallanmaya başlamaktadır.



Şekil 2.4: VF2 algoritmasına göre; G hedef çizgesi üzerinde P sorgu çizgesinin dallanma adımları.

Örnekte "1" sorgu düğümü üzerinden dallanma başlamaktadır. VF2 algoritması "1" düğümünü hedef çizge düğümlerine haritalarken üç kriteri göz önünde bulundurmaktadır:

- Haritalanacak düğümün derecesi en az "1" düğümünün derecesi kadar olmalıdır.

- “1” düğümünün k adet m düğüm derecesine eşit veya büyük dereceye sahip komşusu var ise; haritalanacak düğümün de en az k adet m düğüm derecesine eşit veya büyük dereceye sahip komşusu olmalıdır.
- “1” düğümünün p adet haritalanmış, q adet haritalanmamış komşusu var ise; haritalanacak düğümün de en az p adet haritalanmış, q adet haritalanmamış komşusu olmalıdır.

Bu üç kriter göz önünde bulundurulduğunda; A, H ve F düğümleri, düğüm dereceleri sebebiyle uygun birer aday olamamaktadır. G düğümünün ise biri en az 4 diğeri en az 2 dereceye sahip iki komşusu bulunmadığı için uygun bir aday olamamaktadır. “1” düğümüne uygun adaylar seçildikten sonra dallanma herhangi bir komşusu üzerinde devam eder. Mümkün olan tüm dallanmalar denenip / budandıktan sonra kalan dallanmalardan sorgu çizgesi ile aynı boyuta sahip olanlar uygun birer sonuç teşkil ederler.

Her ne kadar verimli algoritmalar gereksiz dallanmaları olabildiğince erken budasa da algoritma yaklaşımının doğası gereği işlem yükü oldukça fazladır. Bu dezavantajının yanında küçük ve orta ölçekli çizgelerde, herhangi bir önişleme ihtiyaç duymaması ve hafıza tüketiminin oldukça düşük olması sebebiyle hala tercih edilmeye devam etmektedir. Alt-çizge eşlemesi yapılmak istenen çizgelerin boyutu binleri aşmayan, herhangi bir ön işleme ihtiyaç duymayan ve hafıza bakımından kısıtlı sistemlerde rahatlıkla dal-ve-sınır algoritmaları kullanılabilir. Diğer yandan eğer hedeflenen çizgeler oldukça büyük ya da sayısı çok fazla ise indeks tabanlı yaklaşıma sahip bir algoritma tercih edilmelidir.

İndeks tabanlı yaklaşımlar, problemin çözümü için üzerinde en çok çalışılmış ve en çok algoritma üretilmiş çözüm yaklaşımıdır. Bu yaklaşımı kullanan algoritmalar problemi altı temel işlem adımı ile çözmektedir:

- Adım 1: Hedef çizge ya da çizge veri tabanı belirlenen bir kural kümesi kullanılarak alt bölümlerine parçalanır.
- Adım 2: Elde edilen parçalar, daha sonra filtreleme ve doğrulamada kullanılmak üzere etkin bir şekilde erişilebileceği veri yapılarında saklanır.
- Adım 3: İşleme alınan sorgu çizgesi, hedef çizge ile aynı kurallar dizisine göre parçalanır.

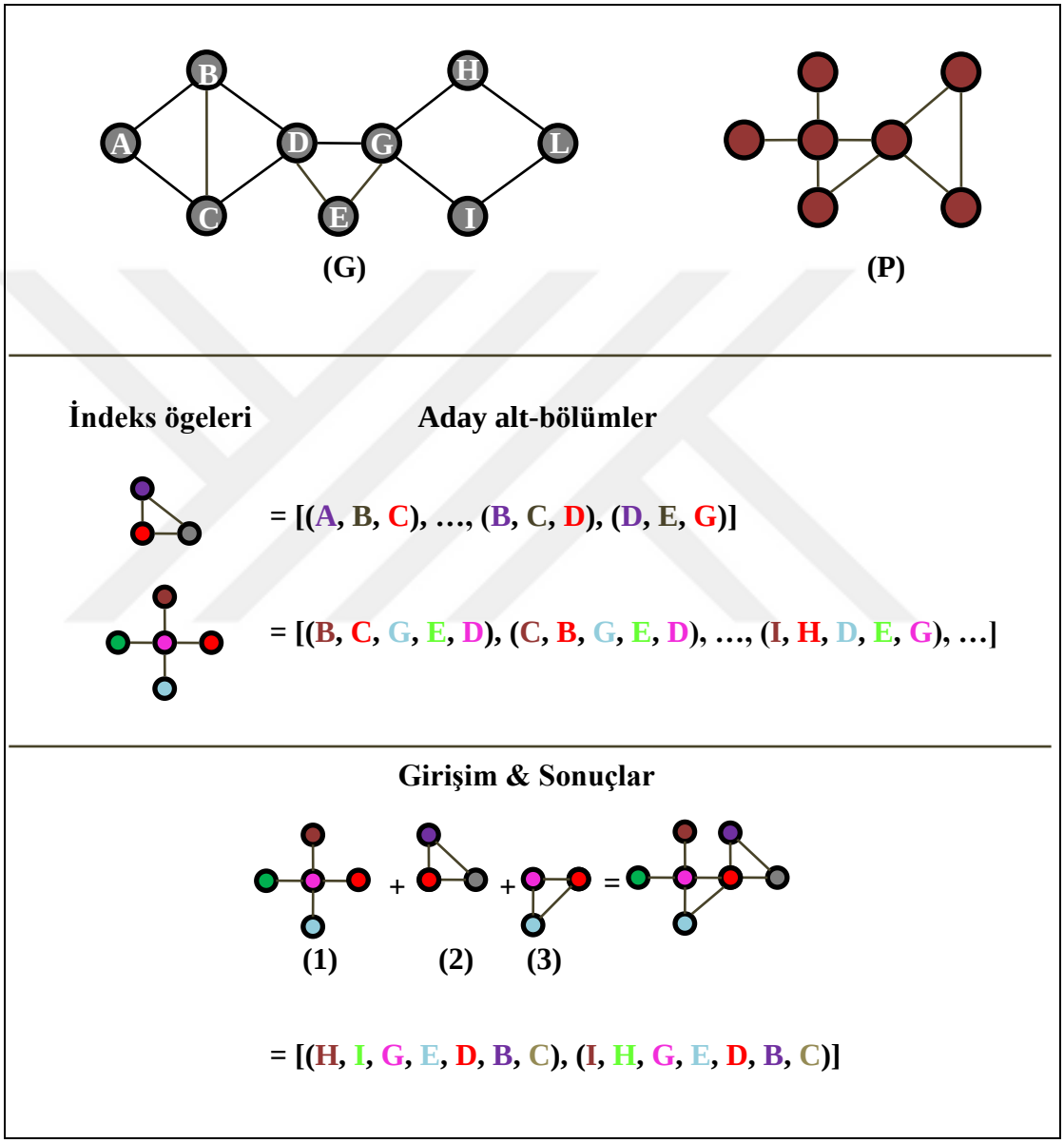
- Adım 4: Hedef çizgenin alt bölümlerinin tutulduğu veri yapıları sorgu çizgesinin alt parçalarına uygun adaylar bulmak için filtrelenir.
- Adım 5: Sorgu çizgesinin alt parçalarına ait aday kümelerinin, Adım 3'te gerçekleştirilen parçalanmanın tersi yönde girişimi ile kısmi sonuçlar üretilir.
- Adım 6: Kısmi sonuçlar eşbiçimlilik kontrolüne tabi tutulur ve nihai sonuçlar elde edilir.

Yukarıda bahsi geçen adımlarda yapılan bir iyileştirme algoritmanın hızını oldukça arttırmaktadır. Araştırmacılar bu adımların optimizasyonu için birçok farklı yöntem geliştirmiştir. Adım 1 de bahsi geçen kural kümesinin belirlenmesi problemi, diğer bir deyişle çizgenin hangi öznitelikleri kullanılarak indeksleneceğinin belirlenmesidir. Bu alt problem araştırmacılar tarafından üzerine en çok düşünülmüş alt problemdir çünkü çizgeler birçok özniteliğine göre indekslenebilmektedir. Bazı yöntemlerde çizgeler yapısal öğelerine göre indekslenmektedir. Bu yapısal öğeler belirli bir uzunluktaki yollar [Zou et al., 2008], [Zhang et al., 2007], [Zhao et al., 2007], ağaçlar [Shang et al., 2008], [Larrosa and Valiente, 2002], [Zampelli et al., 2010] veya alt-çizgeler [Nabti and Seba, 2016], [Zou et al., 2009], [Afrati et al., 2013], [Hong et al., 2015] olabilmektedir. Diğer yandan geliştirilen bazı algoritmalar [Zhang et al., 2010], [Messmer and Bunke, 1998], [Gao et al., 2010] çizgeleri düğüm etiketleri, düğüm ağırlıkları gibi içerik öğelerine göre indekslemektedir.

Şekil 2.5'te görülen örnekte indeks ögesi olarak yıldız ve üçgen alt yapıları kullanılmaktadır. Sorgu çizgesi bu yapılara göre parçalandığında şekildeki örnek aday alt bölümler elde edilmektedir. Hedef çizge indekslendikten sonra sağ üstte gözüken sorgu çizgesi için eşleme işlemi başlamaktadır. Örnekte görülen sorgu çizgesi indeks ögelerine tam olarak ayrılabilmektedir. Daha sonra bu alt yapıların adayları girişim işleminden geçirildiğinde sağ altta görülen nihai sonuçlar elde edilmektedir.

Elde edilen indeks ögelerinin, indekslenen özniteliğe uygun bir veri yapısında tutulması algoritmanın performansının arttırılmasında önemli rol oynar. Örneğin; bir yöntem elde ettiği indeks verilerinden bir önek ağacı [He and Singh 2008] oluşturarak ihtiyaç duyduğu adayları hızlı bir filtreleme işlemi ile elde etmektedir. Hedef çizge parçalanıp indekslendikten sonra algoritmalar herhangi bir sorgu çizgesini cevaplayabilecek hale gelmektedir. Sorgu çizgeleri de belirlenen kural kümesiyle parçalanarak alt bölümlerine ayrılır. Filtreleme işlemi sırasında sorgu

çizgesinin bu alt bölümlerine uygun adaylar indeks veri yapısından seçilir. Uygun adaylar seçildikten sonra sorgu çizgesinin alt bölümlerinin bir araya getirilmesi ile tekrar oluşturulması işlemi başlar. Yeniden oluşturma sırasında sorgu alt parçalarını kullanarak sorgu çizgesinin tekrar elde edilmesi için bir birleştirme stratejisi belirlenir.



Şekil 2.5: Hedef çizgenin alt yapılar ile indekslenmesi ve uygun sonuçların filtrelenerek sorgu cevaplarının üretilmesi.

Daha sonra bu alt parçalara ait aday kümeleri birbirleri ile birleştirme stratejisine göre girişim işlemine tabi tutulur. Bu girişim işlemi sonucunda birçok sonuç niteliği taşımayan ara sonuçlar üretilmektedir. Bu ara sonuçların temizlenmesi

ve nihai sonuçların elde edilmesi için geçici sonuçlar teker teker sorgu çizgesi ile eşbiçimlilik sorgusuna tabi tutulur. Eşbiçimlilik işleminden başarı ile geçen sonuçlar nihai sonuç olarak kaydedilir.

2.3.2. Yaklaşık Eşleme Yaklaşımı

Yaklaşık eşleme yaklaşımı, birebir eşleme yaklaşımdan farklı olarak karşılaştırılan çizgelerin eşbiçimlilik şartlarını birebir sağlamasını beklemediği için bu ismi almıştır. Bu benzerlik şartları ayrıtlar, düğümler ya da komşuluklar gibi yapısal özellikler olabileceği gibi düğüm etiketleri, düğüm-ayrıt ağırlıkları, ayrıt yönleri gibi içerik özellikleri de olabilmektedir. İki çizge arasında yaklaşık eşleme yapabilmek için: Algoritma dikkate aldığı eşleme ögelerinin; eksikliğinin, fazlalığının ya da yanlış eşleşmesinin düzeltilmesi için gerekli maliyeti hesaplar:

Tanım 8 (Çizge Benzetim Uzaklığı): $G_1 = (V_1, E_1)$ ve $G_2 = (V_2, E_2)$ olmak üzere iki çizge verilmiş olsun. G_1 çizgesinin G_2 çizgesi ile eşbiçimli olabilmesi için yapılması gereken çizge operasyonlarının maliyeti toplamına çizge benzetim uzaklığı denir. Çizge operasyonları: Düğüm, ayrıt, komşuluk, etiket ya da diğer çizge niteliklerinin; eklenmesi, çıkarılması ya da değiştirilmesi olabilir. Toplam maliyet hesaplanırken; çizge operasyonları eşit veya farklı ağırlıklar ile çarpılarak genel toplama eklenir ve böylece toplam maliyet hesaplanır. GED (G_1, G_2) olarak gösterilir.

Yaklaşık eşleme algoritmasına, benzerlik için yeterlilik arz eden oranı bir parametre yardımıyla bildirilmektedir. Hesaplanan operasyonların maliyeti eğer belirlenen bir sınırın altında ise karşılaştırılan bu çizgeler yaklaşık eşbiçimlidir:

Tanım 9 (Yaklaşık Çizge Eşbiçimlilik): $G_1 = (V_1, E_1)$ ve $G_2 = (V_2, E_2)$ olmak üzere iki çizge verilmiş olsun. K hiper parametresi iki çizgenin eşbiçimli olabilmesi için tolerans gösterilebilecek en yüksek çizge benzetim maliyeti olsun. Eğer GED (G_1, G_2) $\leq K$ ise G_1 çizgesi ile G_2 çizgesi yaklaşık olarak eşbiçimlidir.

Yaklaşık eşleme yapan algoritmalar da birebir eşleme algoritmaları gibi temelde iki grup altında toplanmaktadır. Bir kısım algoritmalar belirli benzerlikteki

tüm alt-çizgeleri bulmayı hedeflerken bir diğer grup ise sorgu çizgesi ile aralarında en düşük benzetim maliyetine sahip k adet alt-çizge bulmayı hedeflemektedir.

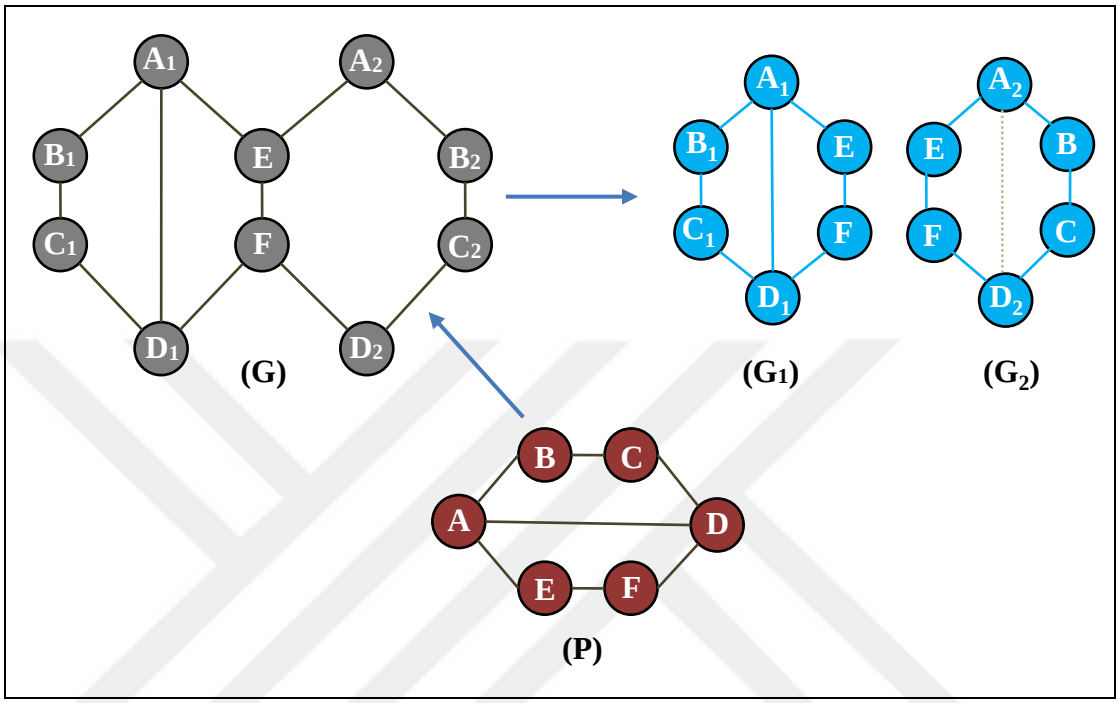
Sınır tabanlı algoritmalar, ismini belirli bir benzerlik değerine sahip hedef alt-çizgeleri bulma stratejisinden almaktadır. Benzerlik değeri hesaplanırken; algoritmalar [Jiang et al., 2007], [Peng et al., 2014] genellikle çizge benzetme uzaklığı (GED) [Gao et al., 2010] gibi genel bir maliyet fonksiyonu ya da kendi yaklaşımlarına uygun bir maliyet fonksiyonu kullanabilmektedir. Örneğin; NSSRF [Zhang et al., 2017] algoritması benzetim maliyetini hesaplamak için kosinüs benzerliğinden yararlanan bir makine öğrenmesi yöntemi kullanmaktadır. Genel benzerlik hesaplama algoritmalarının yanında bazı algoritmalar [Yuan et al., 2012], [Tian and Patel, 2008], [Shang et al., 2010], [Gülsoy and Kahveci, 2011] ise üzerinden yaklaşık eşleme yaptığı özelliğe has bir benzerlik değeri hesaplama yöntemi kullanmaktadır.

Sınır tabanlı algoritmalar, ayrıtların [Mongiovi et al., 2010], [Lacroix et al., 2006], [Zampelli et al., 2005], [Liang and Zhao, 2017], düğümlerin [Willet et al., 1998], [Yuan et al., 2011], komşulukların [Jin and Yang, 2011], belirli uzunluktaki yolların [He and Singh, 2006], [Khan et al., 2011] veya düğümler arası en kısa mesafelerdeki ağırlıklar toplamının [Khan et al., 2013] eksikliğine, fazlalığına veya eşleşmemesine göre bir karşılaştırma yapar ve benzerlik değeri belirler. Hesaplanan bu benzerlik değeri eğer kullanıcı tarafından belirlenen benzerlik parametresi değerinin üzerinde ise bu durumda karşılaştırma yapılan alt-çizge geçerli bir sonuç olarak kaydedilir. Bu yöntem ile yaklaşık eşleme yapan algoritmalar belirlenen benzerlik değeri üzerindeki tüm alt-çizgeleri bulmayı hedeflemektedir:

Tanım 10 (Sınır Tabanlı Yaklaşık Alt-çizge Eşbiçimlilik Problemi): G = (V, E) ve H = (V_2, E_2) olmak üzere iki çizge verilmiş olsun. G' = (V', E'), G çizgesinin herhangi bir alt-çizgesi olsun. M hiper parametresi, iki çizgenin yaklaşık olarak eşbiçimli olabilmesi için tolerans gösterilebilecek en yüksek çizge benzetim maliyeti olsun. G içerisinde bulunan ve GED (G', H) ≤ M koşulunu sağlayan G' alt-çizgelerinin tamamının bulunması problemine sınır tabanlı yaklaşık alt-çizge eşbiçimlilik problemi denir.

Şekil 2.6'da bir yaklaşık alt-çizge eşleme örneği verilmektedir. Örnekte G hedef çizgesinin, P sorgu çizgesi ile en fazla bir ayrıtı farklı (eksik, fazla ya da yanlış

eşleşmiş) olan alt çizgeler aranmaktadır. Örnekte görüldüğü üzere G_1 çizgesi sorgu ile doğrudan birebir eşleşebilmektedir. Diğer yandan, G_2 çizgesi ile P çizgesinin birebir eşleşebilmesi için A_2 ile D_2 arasında bir ayrıta ihtiyaç vardır. Sorgu bir ayrıtın eksikliğine tolerans gösterildiği için G_2 alt-çizgesi de uygun bir sonuç olmaktadır.



Şekil 2.6: G hedef çizgesi ve P sorgu çizgesinin bir ayrıt tolerans ile yaklaşık alt-çizge eşleme sonuçları (G_1 ve G_2).

Yaklaşımın hatalara toleranslı olması; eksik, hatalı veya gürültülü çizgeler içerisinde eksiksiz bir eşleme işlemi uygun hale getirmektedir. Diğer taraftan hali hazırda NP-Tam olan bir problemin arama uzayını daha da arttırdığından çalışma süresi performansı bakımından verimsizdir. Algoritmalar performans sorununu birebir eşleme de olduğu gibi indeks yapıları ile veya dağıtık sistemlerde paralel işlem yürüterek [Peng et al., 2014] aşmaya çalışmaktadır. Çizge boyutları milyonlarının çok büyüdüğü durumlarda bahsi geçen iyileştirmeler de yetersiz kalmaktadır. Bu durumda eğer performans, eksiksiz sonuç üretmekten daha önemli ise K-NN yaklaşımına başvurulmaktadır.

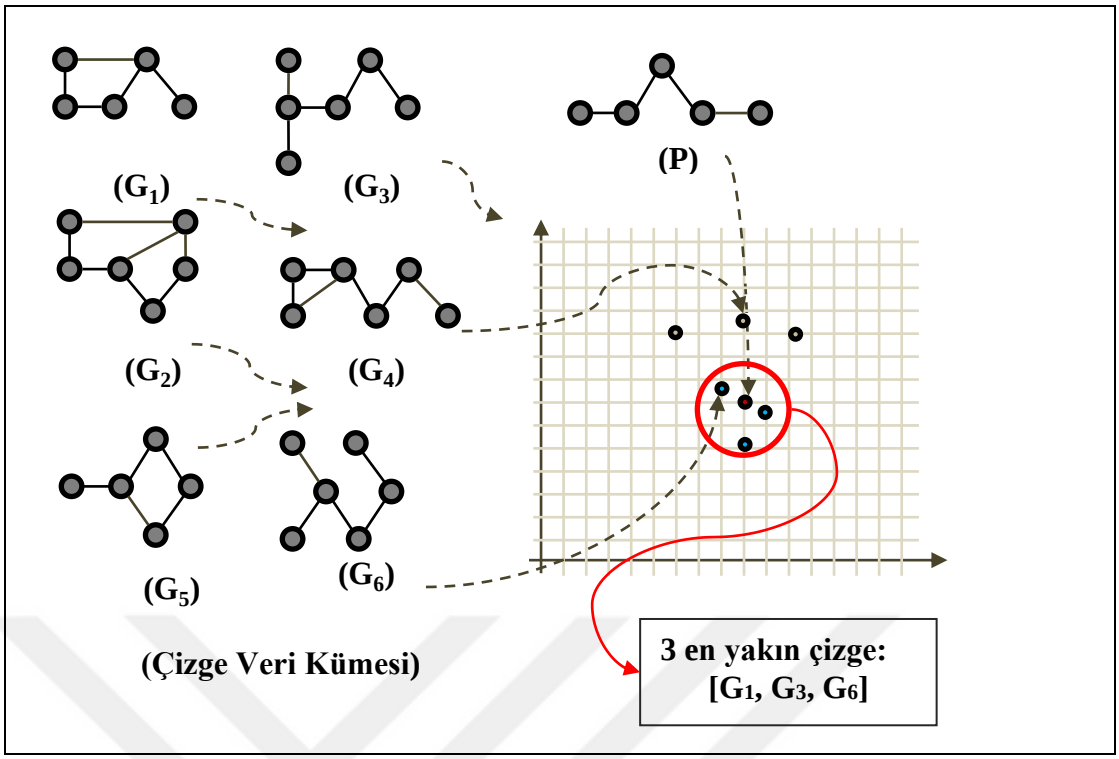
K-NN tabanlı yaklaşımlar, hedef çizge veya çizge veri kümesi içerisinde, sorgu çizge ile en düşük benzetim uzaklığına sahip k adet alt-çizgeyi bulmayı hedeflemektedir (Tanım 11). İsminden de anlaşılacağı üzere bu yaklaşım olası tüm sonuçları bulmak yerine sadece k adet sonuç üretir. Sınır tabanlı yaklaşıma göre

eksik sonuç üretme dezavantajı performans açısından değerlendirildiğinde, K-NN yaklaşımını oldukça üstün kılmaktadır.

Bu yaklaşıma sahip algoritmalar, çizgelere; çizge kapamasına [Pienta et al., 2014], komşuluk matrislerine [Vento, 2015], [Foggia et al., 2014], düğüm-ayrıt özelliklerinin yoğunluklarına [Jin et al., 2013] göre bir değer belirlemektedir. Sorgu çizgesi de aynı değerlendirmeden geçirdikten sonra algoritmanın iki değer arasındaki benzerliği hesaplama yöntemine göre uzaklıklar hesaplanarak en yakın k tanesi sonuç olarak üretilmektedir.

Tanım 11 (K-NN Tabanlı Yaklaşık Alt-çizge Eşbiçimlilik Problemi): $G_1 = (V, E)$ ve $H = (V_2, E_2)$ olmak üzere iki çizge verilmiş olsun. $G_2 = (V', E')$, G_1 in herhangi bir alt-çizgesi olsun. K hiper parametresi, bulunmak istenen yaklaşık eşbiçimli çizge sayısı olsun. G_2 içerisinde bulunan ve GED (G_2, H) uzaklığı en düşük olan K adet G_2 alt-çizgenin bulunması problemine K-NN tabanlı yaklaşık alt-çizge eşbiçimlilik problemi denir.

Şekil 2.7'de çizge veri kümesi ile sorgu çizgesi aynı uzayda vektör olarak ifade edilmektedir. Örnekteki vektör düğüm derece dağılımı vb. yapısal bir öznitelik ile oluşturulmuştur. Tüm hedef çizgeler ve sorgu çizgesi aynı düzlemde ifade edildikten sonra kosinüs uzaklığı vb. bir yöntem ile en yakın k adet çizge bulunmaktadır.



Şekil 2.7: Hedef çizge kümesi ve P sorgu çizgesinin ortak uzayda ifade edilerek k yakın komşusunun bulunması.

K-NN yaklaşımına sahip yöntemler hem tüm sonuçları bulmadığı hem de bulduğu sonuçlar ile sorgu alt-çizgesi arasında belirli bir benzerlik değeri garanti etmediği için kesinlik gerektiren problemlerde kullanılmamalıdır. Öte yandan oldukça büyük çizge veri tabanlarında alt-çizge eşlemesi yaparken veya sadece belirli adet benzer eşbiçimli alt-çizgenin yeterli olduğu: sosyal ağ arkadaş önerisi gibi problemlerde oldukça fayda sağlamaktadır.

Yaklaşık eşleme yöntemlerinden de bahsedildikten sonra problem türleri ve çözümüne yönelik geliştirilmiş tüm yöntem çeşitleri açıklanmış olmaktadır. Sonraki bölümlerde; bu bölümde bahsi geçen problem ve çözüm türleri iki parçalı çizgeler üzerinde ele alınmaktadır. Problemlere bu çizgeler için geliştirilen çözüm yöntemleri detaylıca açıklanmakta ve literatürdeki benzer yaklaşımlar ile performans kıyaslama sonuçları bildirilmektedir.

3. İKİ PARÇALI ÇİZGELERDE BİREBİR EŞLEME

Bu tez çalışması kapsamında iki parçalı çizgeler (Tanım 12) için özelleştirilmiş üç farklı alt-çizge eşbiçimlilik algoritması geliştirilmiştir. Bu algoritmalardan iki tanesi bölüm 2.3.1 de bahsi geçen birebir eşleme problemine, iki parçalı çizgeler üzerinde çözüm üretmektedir.

Tanım 12 (İki Parçalı Çizge): İki parçalı çizge G = (U, V, E), düğümleri ayrık (ortak düğümü bulunmayan) iki kümeye ayrılabilen (U, V kümesi); E kümesindeki her bir ayrıtı, U kümesindeki bir düğüm ile V kümesindeki bir düğümü birbirine bağlayan özel bir çizge türüdür.

İki parçalı çizgelerin eşbiçimlilik durumu test edilirken; haritalanan düğümlerin ait oldukları parçaların da aynı olması, iki uzunlukta komşu olan düğümlerin aynı parçada olması vb. kısıtların da göz önünde bulundurulması gerekir:

Tanım 13 (İki Parçalı Çizge Eşbiçimlilik): G = (U, V, E) ve G' = (U', V', E') olmak üzere iki adet iki parçalı çizge verilmiş olsun. G çizgesindeki her bir düğümü G' çizgesinde bir düğüme; $u \in U$, $v \in V$ ve $e = (u, v) \in E$ iken; $I(u) \in U'$, $I(v) \in V'$ ve $(I(u), I(v)) \in E'$ olacak şekilde haritalayabilecek birebir ve örten bir fonksiyon $I: (V \rightarrow V', U \rightarrow U')$ varsa; G ve G' iki parçalı çizgeleri birbirleri ile eşbiçimlidir.

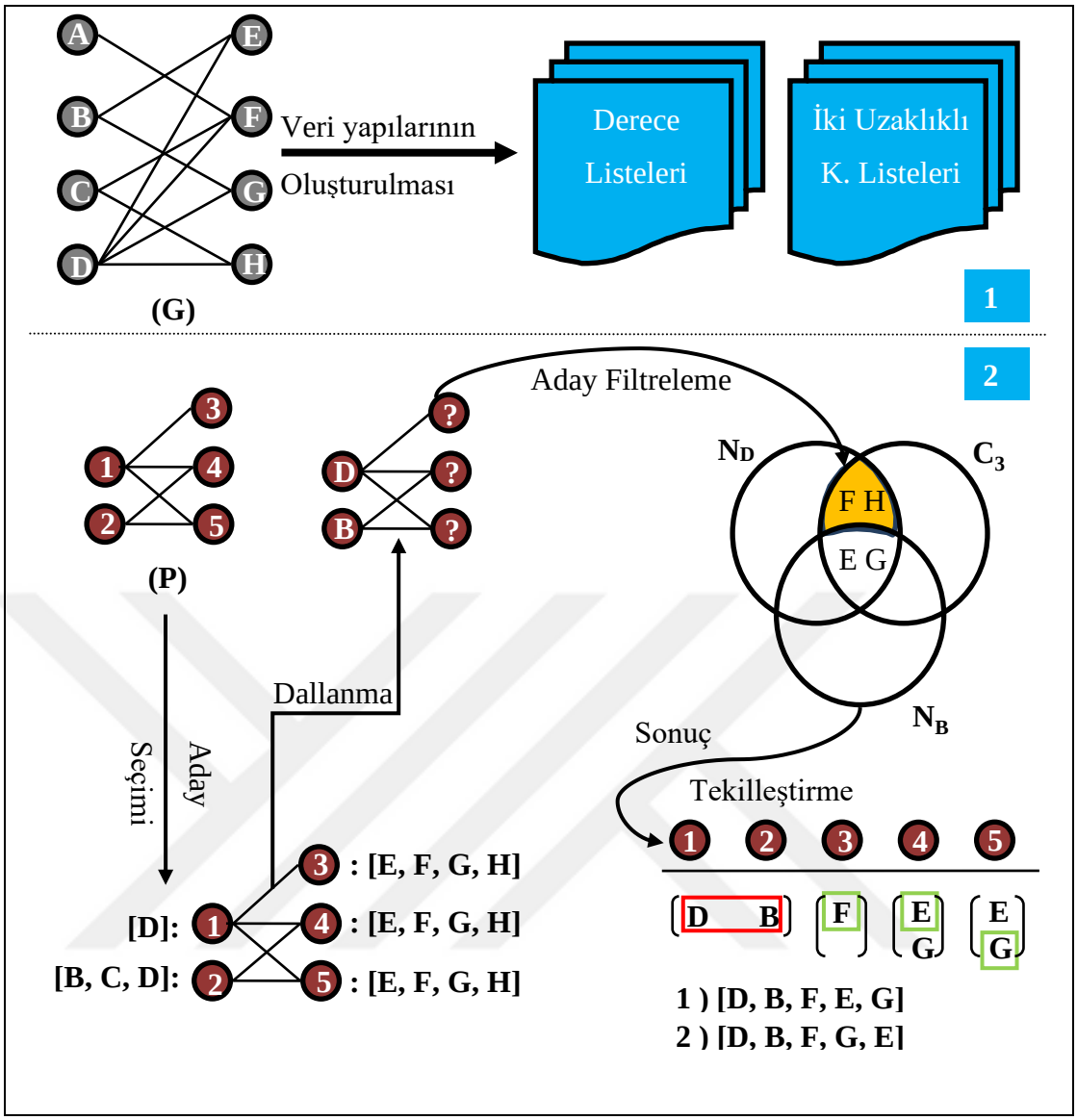
Eşbiçimlilik tanımı, iki parçalı çizgelerde tekrar tanımladığı için iki parçalı çizgelerde alt-çizge eşbiçimlilik probleminin de yeninden tanımlanması gerekmektedir:

Tanım 14 (İki Parçalı Çizgelerde Alt-çizge Eşbiçimlilik Problemi): $G_1 = (U_1, V_1, E_1)$ ve $G_2 = (U_2, V_2, E_2)$ olmak üzere iki adet iki parçalı çizge verilmiş olsun. $|U_1| \leq |U_2|$, $|V_1| \leq |V_2|$, $|E_1| \leq |E_2|$ ve H, G_2 çizgesinin herhangi bir alt-çizgesi olsun. G_2 içerisinde bulunan ve G_1 ile iki parçalı çizge eşbiçimlilik gösteren tüm H alt-çizgelerinin bulunması problemidir.

Probleme bu tür çizgeler üzerinde çözüm getirmeyi amaçlayan bir algoritma için bu kısıtlar, performansı arttırmada oldukça yardımcı olmaktadır. Bahsi geçen kısıtlar kullanılarak: dal-ve-sınır yaklaşımına sahip bir algoritmanın gereksiz dallanmalarının önüne geçilebilmekte veya indeks tabanlı bir algoritmanın etkin bir filtreleme yapılabilmektedir. Bu motivasyonla bu kısıtların etkisini görmek için: Hem dal-ve-sınır tekniğine sahip bir algoritma hem de indeks tabanlı başka bir algoritma geliştirilmiştir. Bölüm 3.1'de dal-ve-sınır tekniğine sahip algoritma sunulduktan sonra bölüm 3.2'de geliştirilen indeks tabanlı algoritma anlatılmaktadır.

3.1. Dal-ve-Sınır Tabanlı Birebir Eşleme Yöntemi

Geliştirilen yöntemde; dallanma, budama ve doğrulama aşamalarında çizgenin iki parçalı yapısı hesaba katılarak performans iyileştirmeleri gerçekleştirilmiştir. İki parçalı yapı sayesinde dallanma uzayı küçülmektedir: Herhangi bir adımda dallanma yapılırken sadece karşı parçadaki düğümler üzerine dallanma gerçekleştirilir. İkili parçanın sağladığı diğer bir kısıt ise; iki uzaklıklı dallanmaların birbiri ile aynı parçada olmasıdır. Bu kısıt sayesinde sadece iki seviye önceki düğümle aynı parçada olan düğümler üzerine dallanma gerçekleştirilebilmektedir. Hem bir hem de iki uzaklıklı komşulukların kontrolü sayesinde algoritma sorgu çizgesi boyutunda bir dallanma ürettiğinde; bu dallanma, doğruluğunun kontrolüne gerek kalmadan uygun bir sonuç olarak kaydedilmektedir.



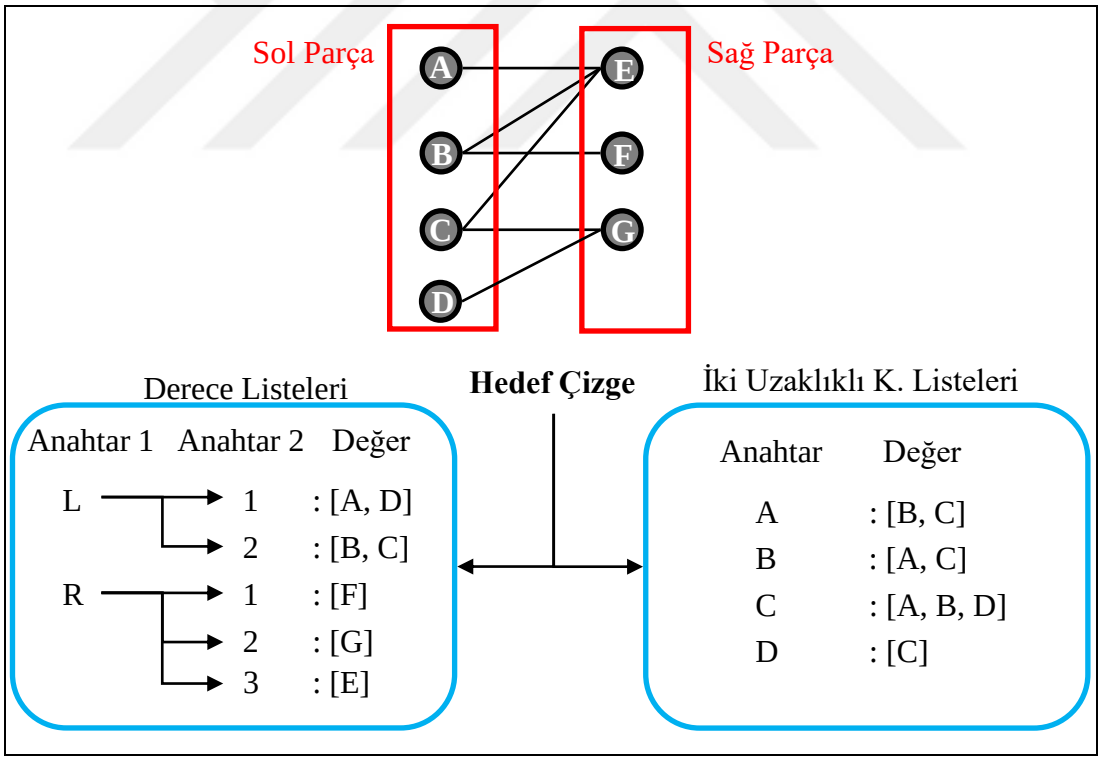
Şekil 3.1: Dal-ve-sınır tabanlı alt-çizge eşleme algoritmasının örnek çalışma adımları.

Bölüm 2'de bahsedildiği üzere dal-ve-sınır algoritmaları üzerinde geliştirmeler temelde üç ana işlem üzerinde yapılmaktadır: Başlangıç düğümünün seçimi, dallanma / budama stratejisi ve üretilen sonuçların doğruluğunun kontrolü. Dallanma ve budama için geliştirilen strateji ile sonuçların doğruluğunun testine gerek kalmaması sayesinde algoritmanın performansı oldukça artmaktadır. Genel işleyişi Şekil 3.1'de görülen algoritma temelde üç adımdan meydana gelmektedir. İlk adımda hem dallanma hem de budama da faydalanılan veri yapıları oluşturulmaktadır. İkinci adım aday seçimi, dallanma ve filtreleme alt adımlarından oluşan eşleme işlemidir. Son adımda, elde edilen çoğul sonuçların tekilleştirilmesi ile nihai sonuçlar elde

edilmektedir. Algoritmanın bahsi geçen tüm bu adımlarına alt başlıklarda detaylı olarak değinilmektedir.

3.1.1. Veri yapılarının üretilmesi

Geliştirilen algoritmada eşleme işlemine başlanmadan önce iki farklı tipte veri, hedef çizgeden toplanmaktadır: *derece listeleri* ve *iki uzaklıklı komşuluk listeleri*. Toplanan veriler, gerek eşleşme ihtimali olmayan dallanmaların budanmasında gerekse oluşan dallanmanın doğruluğunu garanti edilmesinde kullanılmaktadır. Tezin bundan sonrasında anlatım kolaylığı olması amacıyla: çizgenin parçaları sağ ve sol olarak isimlendirilmektedir. Şekil 3.2'de örnek bir hedef çizgeden elde edilen veri yapıları gözükmektedir. Sol altta gözüken veri yapısı *derece listeleri;* hedef çizge düğümlerinin, ait olduğu parça ve düğüm derecesi anahtar olacak şekilde saklandığı bir karma tablosudur.



Şekil 3.2: Budama ve doğrulama için kullanılan veri yapıları.

Bu veri yapısı, sorgu çizgesi düğümlerine aday seçimi yapılırken kullanılmaktadır. Bu veri yapısının oluşturulabilmesi için her bir hedef çizge

düğümünün derecesi bulunmalıdır. Düğüm dereceleri, her bir ayrıtın bir kez ziyaret edilmesi ile bulunabilir ve bu sebeple bu işlem $\Theta(|E|)$ zaman karmaşıklığında gerçekleştirilebilir. Her bir düğüm sadece bir kez saklandığı için $\Theta(|U| + |V|)$ hafıza alanına ihtiyaç duyulmaktadır.

```
# VeriYapılarınıOluşturma (Çizge)
1 Çizgedeki her bir ayrıt için:
2     Ayrıtın sol parçada bağlı olduğu v düğümü iki_uzaklıklı_komşuluk -
      tablosunda anahtar değer olarak yoksa ekle
3     iki_uzaklıklı_komşuluk["v"] listesine ayrıtın sağ parçadaki düğümünün-
      tüm komşularını ekle
4 Çizgedeki her bir düğüm için:
5     Düğüm v'yi derece_listeleri[v düğümünün ait olduğu parça][D(v)] -
      listesine ekle
6 iki_uzaklıklı_komşuluk ve derece_listelerini döndür
```

Şekil 3.3: Veri yapılarını oluşturma sözde kodu.

İki uzaklıklı komşuluk listeleri; sol parçadaki her bir v düğümünün iki ayrıt uzaklıktaki komşu listesinin tutulduğu karma tablolarıdır. Şekil 3.3'teki sözde kodda gösterildiği üzere düğümlerin iki uzaklıklı komşuları, düğümün kimliği anahtar olacak şekilde saklanmaktadır. Bu listeler, sonuç üretmesi imkânsız dallanmaların budanmasını sağlamaktadır: Eğer iki sol düğüm sorgu çizgesi üzerinde birbiri ile iki uzaklıklı komşu ise; haritalanmak istenen hedef çizge adayları da iki uzaklıklı komşuluğa sahip olmalıdır.

Bir düğümün iki uzaklıklı komşulukları bulunurken; çizgenin algoritmaya girdi formatı oldukça önemlidir. Eğer çizge komşuluk matrisi formatında girdi olarak verildi ise; her bir düğümün komşularına ve bu komşuların komşularına $O(1)$ zamanda erişilebilir. Böyle bir durumda toplam kaç kez erişim gerektiğinin hesaplanması gerekir. Sol parçadaki her bir v düğümü için erişim sayısı aşağıdaki formül ile hesaplanabilmektedir:

$$\sum_{i=0}^{|N(v)|} |N(N(v)[i])| \tag{3.1}$$

Formülde $N(v)$ ifadesi v düğümünün komşuluk kümesini, $N(v)[i]$ ise bu kümedeki her bir komşu düğümü ifade etmektedir. Her *v* düğümü en kötü durumda tüm sağ parça düğümleri ile komşu olabilir: En kötü durumda $|N(v)| = |V|$ olmaktadır. *V* düğümünün her bir komşusu, en kötü durumda tüm sol parça düğümleri ile bağlı olabilir: $|N(N(v)[i])|$ en kötü durumda $|U|$ olmaktadır. Bu durumda her bir *v* düğümü için $O(|V| \; x \; |U|)$ erişim gerekmektedir. Bu sebeple iki uzaklıklı komşuluk listelerinin oluşturulması $|U|$ x $O(|V| \; x \; |U|)$ x $O(1) = O(|V| x \; |U|^2)$ zaman karmaşıklığında gerçekleştirilir. Veri yapısının hafıza gereksinimini hesaplamak için: Her bir sol parça düğümünün en fazla kaç iki uzaklıklı komşusu olabileceği belirlenmelidir. Zaman karmaşıklığı hesaplanırken bahsedildiği üzere: her bir sol parça düğümü ile tüm sağ parça düğümleri arasında bir ayrıt bulunabilir. Bu durumda her bir *v* düğümünün $|U| - 1$ adet iki uzaklıklı komşuluğu bulunabilir. Bu durumda veri yapısının saklanması için $O\big(|U|x \; (|U| - \; 1)\big) = O(|U|^2)$ hafıza alanı gerekmektedir.

3.1.2. Dallanma, Eşleme ve Tekilleştirme

Algoritma, dallanma işlemini sorgu düğümlerinin aday listeleri üzerinde gerçekleştirmektedir: Veri yapıları oluşturulduktan sonra her bir sorgu düğümüne, hedef çizge üzerinde haritalanabileceği düğümlerden oluşan bir aday listesi tanımlanır. Verilen bir sorgu çizgesi $P = (U', V', E')$'nin düğümlerine, $G = (U, V, E)$ hedef çizgesi içerisinden seçilebilecek aday listesi C(u) 3.2'de gösterilen eşitlik ile elde edilebilir:

$$\begin{aligned} &\forall u' \in U';\; C(u') = \{u \in U : D(u) \geq D(u')\}, \\ &\forall v' \in V';\; C(v') = \{v \in V : D(v) \geq D(v')\} \end{aligned} \tag{3.2}$$

Eşitlik incelendiğinde sorgu çizgesinin düğümlerine ait aday kümeleri oluşturulurken hedef çizge düğümlerinin düğüm derecelerinden ve ait oldukları parça bilgisinden faydalanıldığı görülmektedir. Sorgu çizgesindeki bir *v'* düğümünün aday

listesi, düğüm derecesi en az D(*v'*) kadar ve *v'* düğümü ile aynı parçada olan hedef çizge düğümlerinden oluşmaktadır. Her bir sorgu düğümü için aday kümeleri seçildikten sonra algoritma dallanma işlemini başlatır. Dallanma işlemi sorgu çizgesi girdisinde bulunan ilk sol parça düğümü ile başlamaktadır. Bu düğüme, düğümün aday listesindeki ilk hedef düğüm atanır ve sorgu çizgesinin ikinci sol düğümüne geçilir.

İkinci düğüm ataması yapılırken; öncelikle bu düğüm ile birinci düğüm arasında, sorgu çizgesine göre, iki uzaklıklı komşuluk ilişkisi olup olmadığı kontrol edilir. Eğer düğümler arasında iki uzaklıklı komşuluk ilişkisi yoksa: Aday listesindeki ilk aday atlanarak bir sonraki düğüme geçilir. Diğer yandan, birinci düğüm ile iki uzaklıklı komşuluk ilişkisi varsa: Aday listesinde birinci düğümle iki uzaklıklı komşuluğu bulunan ilk aday atanmaktadır. Bu kontrol işlemi için; önişlem aşamasında hazırlanan *iki uzaklıklı komşuluk listelerinden* faydalanılmaktadır. İkinci düğüme de atama yapıldıktan sonra bir sonraki sol düğüme geçilir ve ikinci düğüm için gerçekleştirilen aday atama prosedürü tekrar çalıştırılır. Burada dikkat edilmesi gereken durum: üçüncü düğümün iki uzaklıklı komşuluk kontrolü yapılırken hem ilk hem de ikinci düğümle iki uzaklıklı komşuluk ilişkisinin kontrol edilmesi gerektiğidir. Bu durumu genellemek gerekirse: Her bir sol düğüme aday atanırken (dallanma gerçekleştirilirken); hali hazırda aday ataması yapılmış tüm sol parça düğümleri ile iki uzaklıklı komşuluk kontrolünün yapılması gerekmektedir.

Şekil 3.4'te sözde kodu verilen algoritma incelendiğinde, her bir sol parça düğümüne aday ataması yapıldıktan sonra dallanmanın ikinci aşamasının başladığı görülmektedir. Sağ parça düğümlerine uygun dallanmaları gerçekleştirmek için; her bir sağ parça düğümünün aday listesi bir filtreleme işleminden geçer ve filtreleme işlemi sonunda kalan tüm adaylar, her biri ayrı ayrı, mevcut sol parça dallanması için uygun adaylardır. Bir *v* düğümü için filtreleme işlemi yapılırken öncelikle düğüm ile bağlı (arasında ayrıt bulunan) sol parça düğümlerine atanan adayların komşuluk kümelerinin kesişimi alınarak S_L kümesi elde edilir. S_L kümesi, mevcut dallanma için v düğümünün haritalanabileceği adayları kapsayan en geniş kümedir. Bu durumun sebebi: Aralarında iki uzaklıklı komşuluk ilişkisi olan düğümleri bağlayan düğümün, her iki sinin de komşuluk kümesinde yer alması gerekliliğidir.

```
# Dallanma_ve_Filtreleme (derece_listeleri, iki_u._k., sorgu_çizgesi)
1  Nihai_sonuçlar = [ ]
2  Her bir sorgu çizgesi düğümü v için:
3      derece_listeleri[parça(v)] içerisindeki her bir düğüm_derecesi için:
4          Eğer düğüm_derecesi >= D(v):
5              derece_listeleri[parça(v)][düğüm_derecesi] içindeki tüm hedef
               çizge düğümlerini v düğümünün aday kümesine ekle
6  İşaretçiyi tüm sol parça düğümlerinin ilk adaylarını gösterecek şekilde ayarla
7  İşaretçi aday kümelerinin tamamının son elemanını gösterinceye kadar:
8      Mevcut dallanmadaki adayların iki uzaklıklı komşuluklarını kontrol et
9      Eğer iki uzaklıklı komşuluk ilişkilerinde sorun yoksa:
10         Her bir sağ parça düğümü u için:
11             u düğümünün adayları ∩ u düğümü ile aralarında ayrıt bulunan sol
               parça düğümlerinin komşuları
12             u düğümünün adayları / u düğümü ile aralarında ayrıt bulunmayan
               sol parça düğümlerinin komşuları
13         mevcut_sonuçlar = Tekilleştirme (f_aday_listeleri, tekil_adaylar)
           mevcut_sonuçları Nihai_sonuçlar listesine ekle
14     Olası tüm permütasyonları oluşturacak şekilde işaretçiyi ilerlet
15 Nihai_sonuçlar listesini döndür
16
```

Şekil 3.4: Dallanma ve filtreleme algoritmasının sözde kodu.

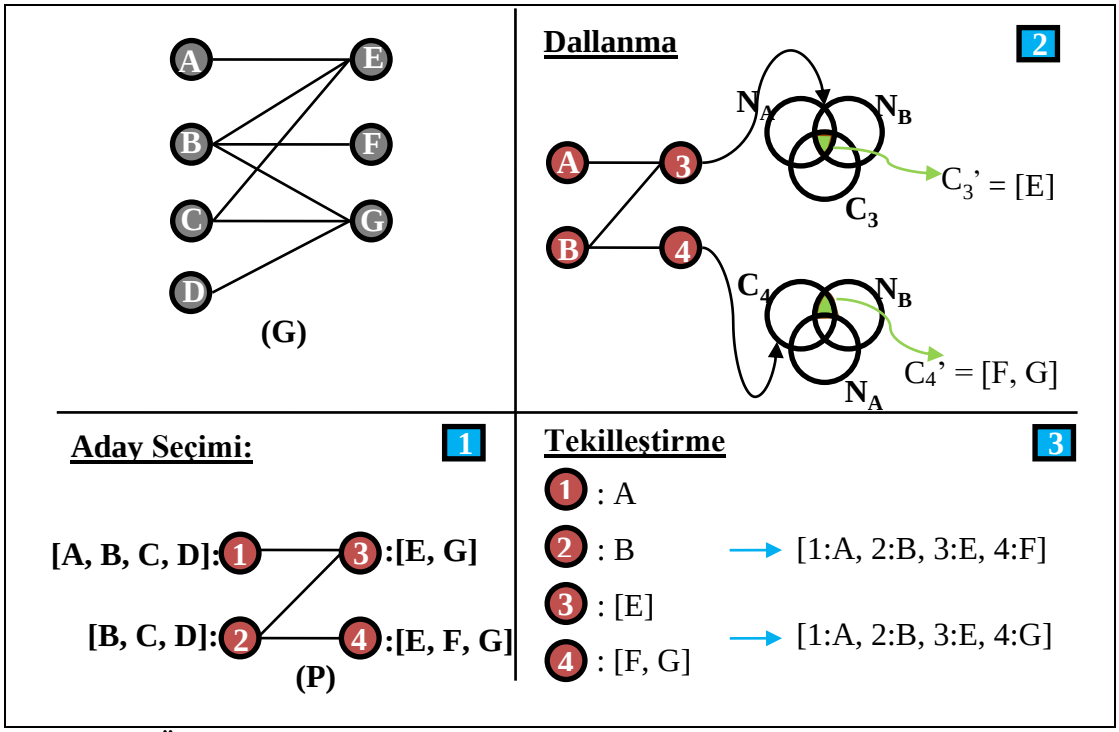
Filtrelemenin bu noktasında eşleme yapılmak istenen hedef alt-çizgelerin indüklenmiş alt-çizgeler (Tanım 15) olup olmamasına göre ekstra bir kesişim işlemi daha gerekmektedir.

Tanım 15 (İndüklenmiş Alt-çizge): G = (V, E) çizgesi verilmiş olsun. S ⊂ V olacak şekilde G çizgesinin düğümlerinin bir alt kümesi olsun. G[S] indüklenmiş alt-çizgesi, düğüm kümesi S, ayrıt kümesi ise E kümesindeki her iki ucu da S kümesinde bir düğüm olan tüm ayrıtların oluşturduğu E' olan çizgedir.

S_L içerisindeki bazı adaylar, sorgu çizgesi üzerinde komşu olmamasına rağmen mevcut dallanmadaki bazı adaylar ile aralarında komşuluk ilişkisi bulunmaktadır. İndüklenmiş alt-çizgelere uygun olmayan adayların elenebilmesi için: Sorgu çizgesi üzerinde *v* düğümü ile komşu olmayan düğümlere atanan adayların komşuluk kümelerinin birleşimi alınarak S_D kümesi oluşturulmaktadır. Bu birleşimden elde edilen küme, *v* düğümünün, mevcut dallanma için haritalanamayacağı hedef çizge düğümlerinin tamamını içermektedir.

Her bir sağ parça düğümü *v*'nin nihai adaylarını elde etmek için öncelikle *v* düğümünün aday kümesi ile S_L kümesinin kesişimi alınır. Bu kesişim işlemi S_L içerisinde derece bakımından uygunsuz adayların elenmesini sağlar. Daha sonra eğer indüklenmiş alt-çizgeler aranıyorsa kalan adaylar ile S_D kümesinin farkı alınır ve nihai adaylar elde edilir. Aksi durumda bu kesişim işlemine ihtiyaç yoktur.

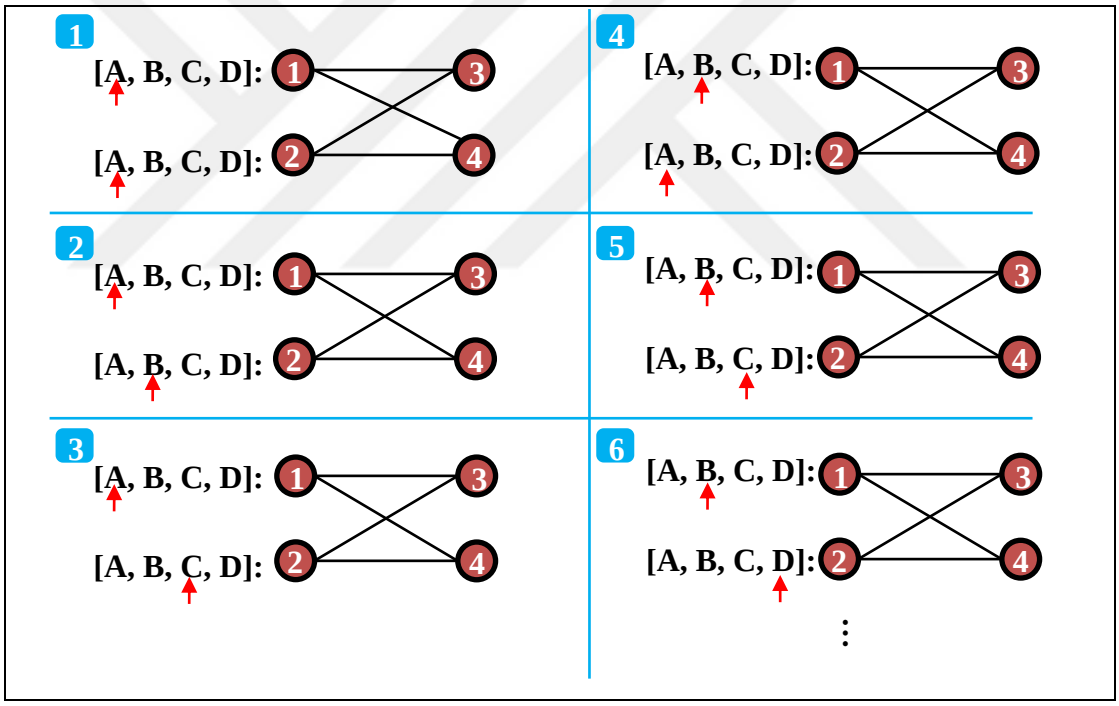
Şekil 3.5'te algoritmanın örnek bir dallanma işlemi ve elde edilen sonuçlar gösterilmektedir. Sağ üst (2) kısımda görüldüğü üzere; her sağ düğüm için kesişim ve fark işlemleri yapılarak filtrelenmiş nihai aday kümeleri elde edilir. 3. kısımda görülen nihai listelerde sol parça adayları tekil iken sağ parça adayları listelerden oluşmaktadır.



Şekil 3.5: Örnek bir G hedef ve P sorgu çizgesi için: aday seçimi, dallanma, eşleme ve tekilleştirme adımları.

Nihai kümelerdeki her bir aday, mevcut sol parça dallanması için, ait oldukları düğüme uygun birer adaydır. Bu aday kümelerinden nihai tekil sonuçları elde etmek için: sağ parça düğümlerinin aday kümelerinin Kartezyen çarpımı alınır. Daha sonra şekilde mavi oklar ile gösterilen tekil sonuçları elde etmek için tekil sol parça düğüm adayları ile birleştirilirler.

Tekil sonuçlar elde edilip saklandıktan sonra, algoritma yeni sonuçlar üretmek için: Sol parçadaki düğümlere sıradan yeni bir aday seçerek devam eder. Bu işlem sol parça düğümlerine ait aday listelerindeki uygun tüm dallanmalar yapılana kadar devam eder. Her bir adımda; aday kümelerindeki adaylar, listelerde belirme sıralarına göre seçilere dallanmalar oluşturulur. Listelerdeki adayların oluşturabileceği tüm dallanmaların oluşturulduğundan emin olmak için bir işaretçi yapısından faydalanılmaktadır.

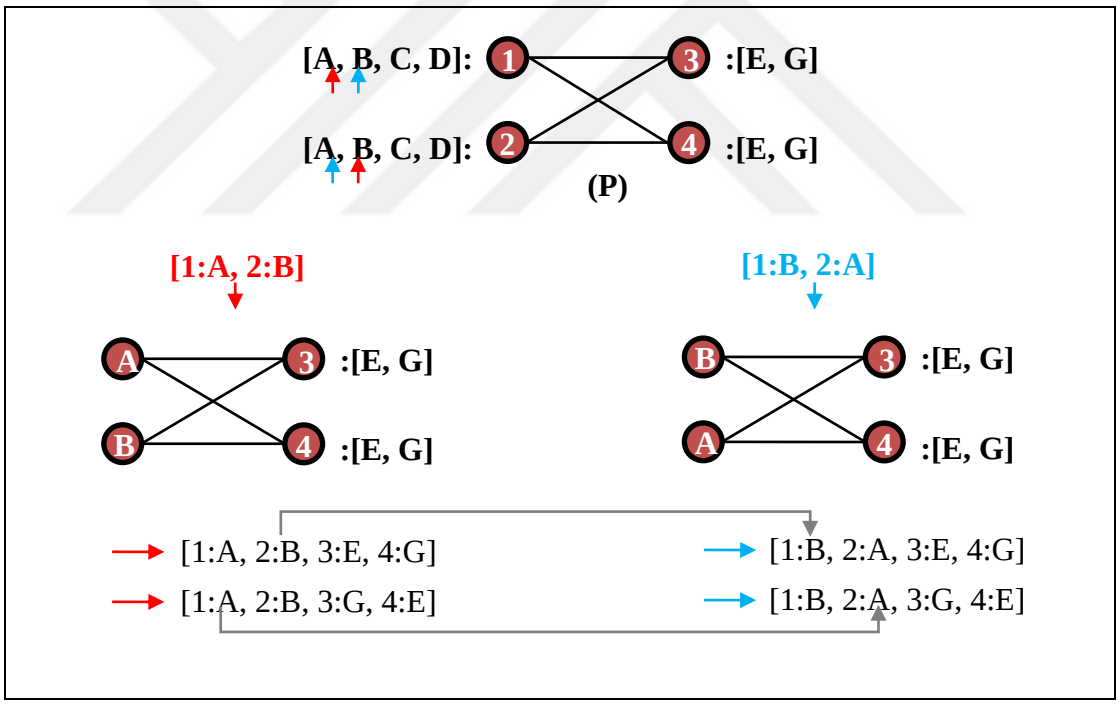


Şekil 3.6: Dallanma için kullanılan işaretçi yapısı.

Şekil 3.5'te verilen hedef ve sorgu çizgesine göre dallanma için kullanılan işaretçi Şekil 3.6'da gösterilmektedir. Şekilde (1) görüldüğü üzere işaretçi ilk başta tüm listelerin ilk adaylarını göstermektedir. İlk uygun adaylar (2) seçildikten sonra, seçilen adayların iki uzaklıklı komşuluğa sahip olup olmadığına kontrol edilerek dallanma budanır veya eşleme işlemine geçilir. Örneğin; düğüm 1'e A, düğüm 2'ye

de D atandığında (4) bu dallanma budanmaktadır çünkü A ile D arasında iki uzaklıklı komşuluk ilişkisi bulunmamaktadır. Geliştirilen işaretçi yapısı öncelikle en sondaki düğümün aday kümesinde ilerlemektedir. İşaretçi bir listenin sonuna geldiğinde (4); bir önceki düğümün aday listesindeki sonraki adaya geçer. Daha sonra sonuna gelinen düğüm ve sonrasındaki düğümlerin işaretçileri, kendi listelerinin başına ayarlanır.

Geliştirilen işaretçi mekanizması sayesinde listelerden seçimler sıralı olarak yapılmaktadır. Bu sayede adayların tüm kombinasyonları eksiksiz ve tekrarsız olarak oluşturulmaktadır. Diğer yandan geliştirilen yöntem yapısal eşleme yapan bir yöntem olduğu için sorgu çizgesi üzerindeki yapısal tekrarlanmalar da göz önünde bulundurulmalıdır: Sorgu çizgesinin sol parçasındaki iki düğüm eğer ikiz düğüm (Tanım 16) ise; bu düğümlere yapılan [a, b] ile [b, a] aday ataması birebir aynı işlemlerin yerine getirilmesine sebep olacaktır.



Şekil 3.7: Şekil 3.5'te gösterilen hedef çizge üzerinde P sorgu çizgesinin ikiz düğümleri ile üretilen ikiz sonuçlar.

Tanım 16 (İkiz Düğümler): G = (V, E) çizgesi verilmiş olsun. u, v ∈ V olmak üzere bu çizgeye ait herhangi iki düğüm N(u) = N(v) koşulunu sağlıyorsa; bu düğümler ikiz düğüm olarak adlandırılmaktadır.

Eğer iki düğümün komşulukları birebir aynı ise geliştirilen algoritmada birebir aynı adayların seçimine ve aynı filtrelemelerin uygulanmasına sebep olacaktır. Şekil 3.7'teki sol düğümler ikiz düğümdür. Düğüm 1 ve 2'nin bu yapısal benzerliği sebebiyle: Düğüm 1'in A, düğüm 2'nin B üzerine haritalanması ile düğüm 2'nin A, düğüm 1'in B üzerine haritalanması sonucu sağ parça düğümlerinin filtrelenmiş aday listeleri arasında fark olmadığı görülmektedir. Bu sebepten ötürü, oluşturulan nihai sonuçlarda sadece A ve B düğüm değerlerinin yer değiştirdiğini, bunun dışında tamamı ile aynı sonuçların üretildiği görülmektedir. Bu durum, algoritmanın gereksiz işlem yapmasına ve dolayısıyla performans kaybına sebep olmaktadır.

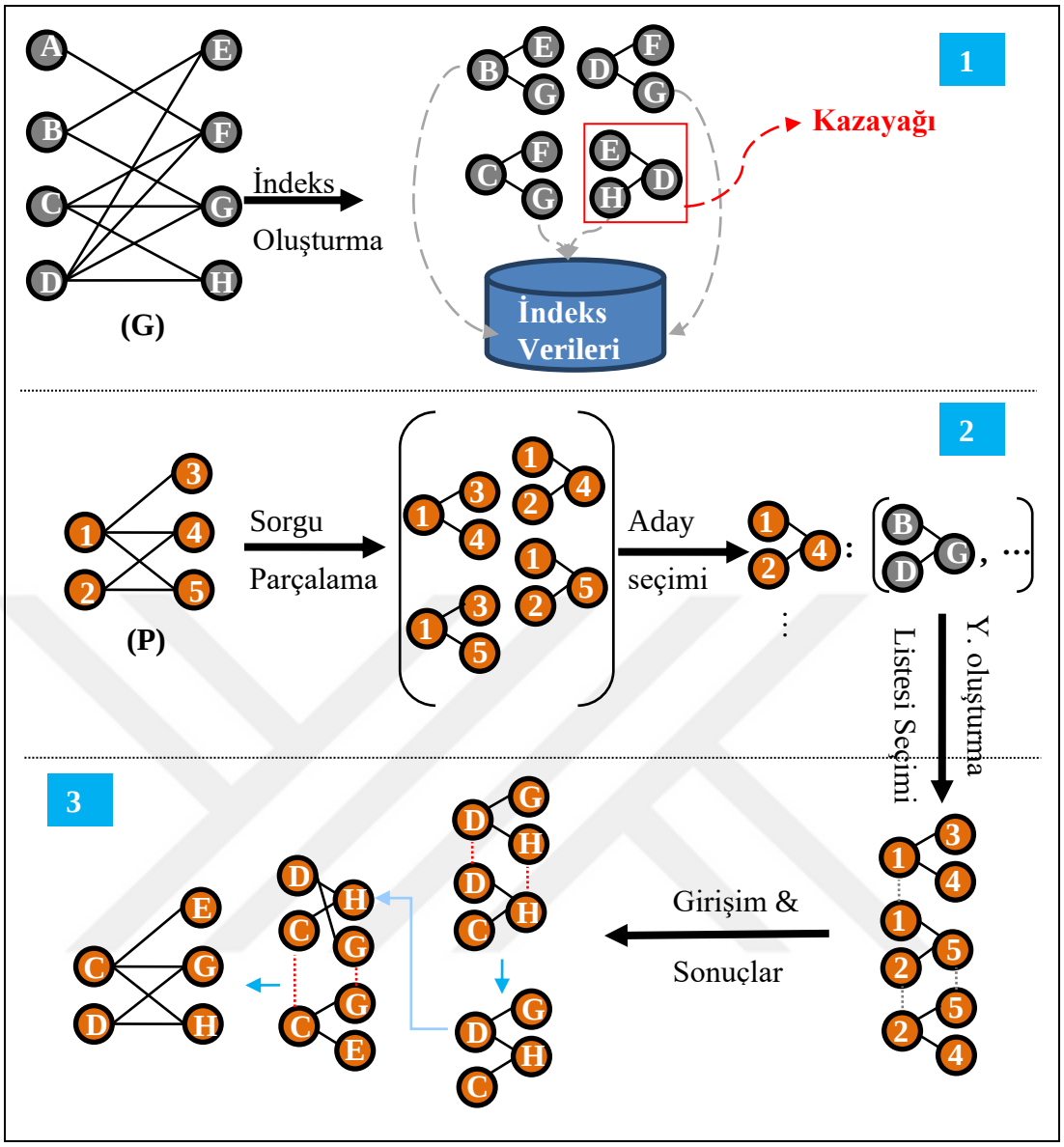
İkiz düğümlerin sebep olduğu tekrarlı hesaplamaları önlenerek algoritmanın performansı arttırılabilmektedir. Bu tekrarlamaların engellenebilmesi için: Algoritma bir dallanma oluşturduğunda öncelikle bu dallanmanın tüm kombinasyonlarının daha önceden yapılmış aramalar içerisinde bulunup bulunmadığını kontrol etmesi gerekmektedir. Bu işlem için: Hızla büyüyen dallanma kümesi içerisinde, sorgu çizgesinin k ikiz düğümü için $k!$ adet arama işlemi anlamına gelmektedir. Bu işlem oldukça maliyetli olmasının yanında getireceği performans kazancından çok daha fazla maliyete sebep olmaktadır. Bu maliyetli işlemden, algoritmanın kullandığı işaretçi yapısı üzerinde yapılacak akıllıca bir değişiklikle kurtulmak mümkündür.

Daha önceki dallanmalar içerisinde bir dallanmanın mevcudiyetini aramaktansa o dallanmanın hiçbir zaman üretilmeyeceğini garantilemek çok daha maliyetsiz bir işlemdir. Bu işlem için dallanma işlemi başlamadan önce ikiz düğüm kümeleri bulunmalıdır. İşaretçi, bir düğüm için aday değişimi yapacağı zaman; düğümün kendinden sonraki düğümler arasında bir ikizi olup olmadığını kontrol etmektedir. Kendinden sonra bir ikiz düğümün olması durumunda, bu ikiz düğümün işaretçisini kendinden bir sonraki indekse ayarlamaktadır. Bu düşük maliyetli işlem sayesinde; diğer düğümlerin dallanma adayları aynı iken ikiz düğümlerin adaylarının kombinasyonu olan dallanmalar sadece bir kez yapılmaktadır. Bu mantığın çalışmasına izin veren durum ise ikiz düğümlerin aday listelerinin daima birbiri ile bire bir aynı ve tekrarsız dizilime sahip olmasıdır. Bu durumun sağlanması için hiçbir ek işleme gerek yoktur çünkü aday listeleri oluşturulurken derece listelerinden faydalanılmaktadır. Bu listeler algoritma dallanma işlemine başlanmadan sadece bir kez oluşturulduğu için bütün düğümler için aynı sıralamadadır.

3.2. İndeks Tabanlı Birebir Eşleme Yöntemi

Geliştirilen indeks tabanlı yaklaşım, tek bir büyük çizge içerisinde yapısal ögeleri indekslemekte ve bu indeks verilerini kullanarak birebir alt-çizge eşbiçimlilik problemini çözmektedir. Bölüm 2'de değinildiği üzere indeks tabanlı algoritmaların çok iyi tasarlanması gerekmektedir. İyi tasarlanmamış bir indeks tabanlı algoritma, hafıza taşmasına, çok uzun indeksleme sürelerine veya kabulü mümkün olmayan filtreleme ve yeniden oluşturma işlemlerine sebebiyet verebilmektedir.

Yapılan bu tez çalışmasında, literatürde bulunan indeks tabanlı yaklaşıma sahip alt-çizge eşbiçimlilik algoritmaları incelenerek üstesinden gelinmesi gereken ortak alt problemler belirlenmiştir. Bu problemlerden ilki hedef çizgeleri en iyi şekilde ifade edebilecek ve aynı zamanda sorgu çizgelerini de karşılayabilecek indeks elemanlarının bulunmasıdır. İndeks elemanları oldukça önemlidir: İyi tasarlanmamış bir indeks yapısı ile sorgu çizgelerini karşılayacak indeks ögeleri bulunamayabilir, aynı veriler tekrar tekrar indekslenebilir ya da hedef çizge indekslenirken bilgi kaybına uğrayabilir. Diğer bir alt problem ise indeks verilerinin saklanacağı veri yapısının oluşturulmasıdır. İndeks verilerinin tutulduğu veri yapıları, algoritmanın filtreleme yaklaşımına ve indeks yapısına uygun olmalıdır. Yaklaşıma uygun olmayan bir veri yapısı, indeksleme ve filtreleme işlemlerini oldukça yavaşlatabilir ya da büsbütün çalışması imkânsız hale getirebilir. Son önemli alt problem, indeks verilerinden sorgu çizgesinin yeniden elde edilmesi işlemidir. Bu aşamada genellikle maliyeti yüksek küme işlemleri (kesişim, birleşim vb.) gerçekleştirilmektedir. Bu aşamada yapılacak bir performans iyileştirmesi algoritmanın genel performansını oldukça arttırmaktadır.



Şekil 3.8: İndeks tabanlı eşleme algoritmasının genel işleyiş adımları.

Yapılan çalışmada, bahsi geçen tüm bu problemlere iki parçalı çizgelerin karakteristik özellikleri göz önünde bulundurularak çözüm yöntemleri geliştirilmiştir. Şekil 3.8'de algoritmanın genel işleyişi anlatılmaktadır. İlk olarak 1 numaralı bölümde görüldüğü üzere hedef çizge, indeks ögelerine (kazayağları) ayrılarak bir veri yapısında saklanmaktadır. Daha sonra gelen sorgu çizgesi de kendini oluşturan kazayağlarına parçalanmakta ve bu kazayağlarına uygun adaylar hedef çizge veri kümesi filtrelenerek elde edilmektedir. Sorgu çizgesinin yeniden oluşturulması için ihtiyaç duyulan kazağıların optimal listesi bulunur ve son olarak bu listedeki kazayağlarının girişimi ile sonuçlar elde edilmektedir. Bahsi geçen tüm bu işlemlere

zorlukları ve bu zorluklara geliştirilen çözüm yöntemleri ile birlikte alt başlıklarda detaylı olarak değinilmektedir.

3.2.1. İndeks ve Veri Yapısının Belirlenmesi

Çizgeler temelde iki yapısal öğeden oluşmaktadır; düğümler ve ayrıtlar. Yapısal eşleme yapan herhangi bir algoritma sadece bu iki öğe ve bu öğelerden elde edilebilen bilgileri indeks verisi olarak kullanabilmektedir. Literatürdeki yapısal eşleme yapan indeks tabanlı çalışmalar incelendiğinde; belirli uzunluktaki yollar, üçgenler, ağaçlar, daha küçük çizgeler gibi birçok indeks yapısı olduğu görülmektedir. Çalışmanın ana odak noktası iki parçalı çizgeler olduğu için bu çizgelere en uygun indeks yapısı seçilmelidir. Bu çizgelerde iki uzunluklu yollar, bir diğer adı ile kazayağları, özel bir yapı oluşturmaktadır. Kazayağları Şekil 3.8'de görüldüğü üzere iki parçalı çizgelerde kaz ayağı şeklini alan iki uzunluklu yollardır. Bu 'V' yapısı, iki parçalı çizgelerin parçalı yapısını etkin biçimde kullanma şansı sunmaktadır: Kazayağların iki ayağındaki düğümler aynı parçada, tepe noktasındaki düğüm ise bu iki düğümden farklı olarak diğer parçada olması gerekmektedir. Bu bilgi sayesinde algoritma birçok alt problemine çözüm üretmede iyileştirmeler sunmaktadır. Örneğin, bir kazayağın ait olduğu parçaya istinaden; eğer tepe noktası sağ parçada olan bir kazayağı ise aday seçiminde tepe düğümü sol parçada olan tüm kazayağları maliyetsiz bir biçimde elenmektedir.

Genel işleyişin anlatıldığı şekle bakıldığında sorgu çizgesinin (P) kazayağları vasıtasıyla kolaylıkla temsil edilebildiği görülmektedir. Algoritma; herhangi bir kısıt uygulamadan, tüm iki uzunluklu yolları indekslediği için hedef çizgede herhangi bir bilgi kaybı olmamaktadır.

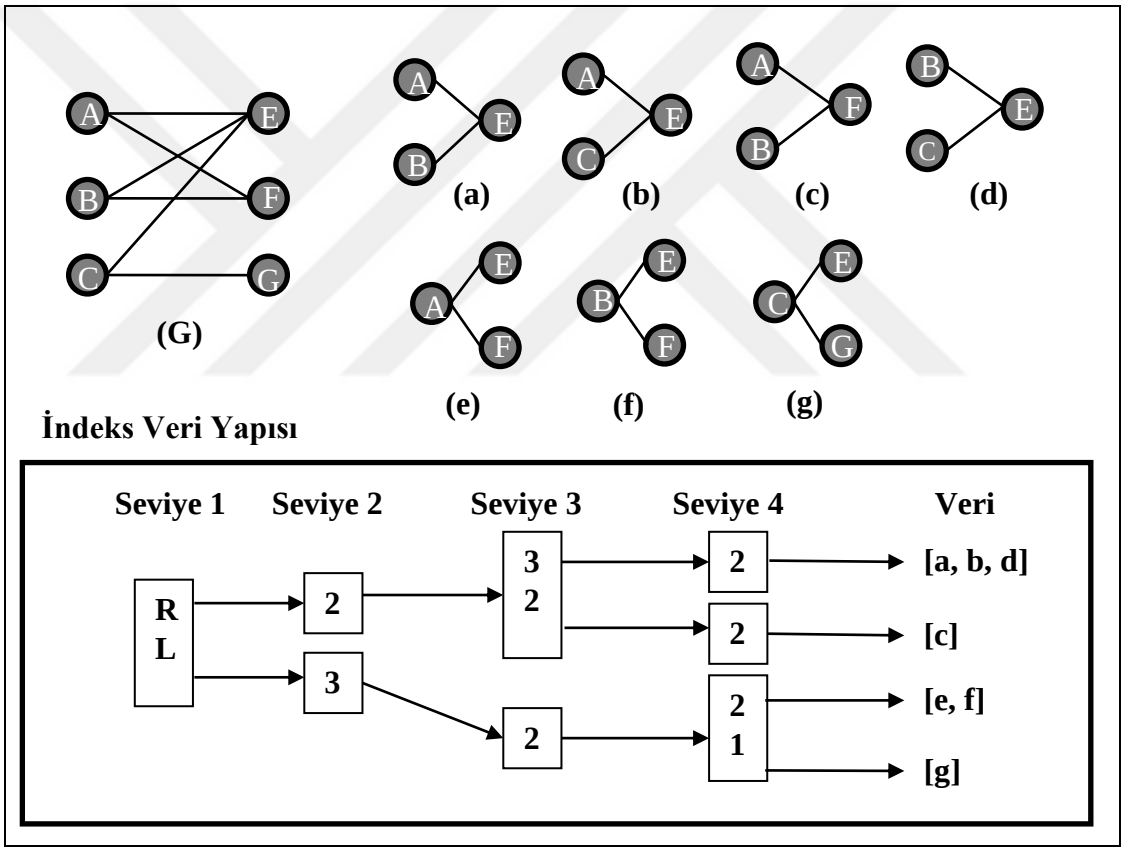
İki parçalı çizgeden elde edilebilecek kazayağları kümesinin ({T}) en büyük eleman sayısı aşağıdaki formülle hesaplanabilmektedir:

$$|\{T\}| = |U| * \binom{|V|}{2} + |V| * \binom{|U|}{2} \tag{3.3}$$

Formül göz önüne alındığında, eşit sayıda düğüm içeren parçalardan oluşan iki parçalı bir çizge için $O(n^3)$ adet kazayağın indekslenmesi gerekmektedir. Binlerce düğümden oluşan çizgeler hedeflendiği için bu sayı milyarlar seviyesindedir. Sorgu

çizgesine ait bir kazayağına aday seçimi yapılırken bunca kazayağın içerisinden uygun adayların filtrelenmesi maliyetli bir işlemdir. Kazayağları, sırasız bir liste veri yapısında tutulduklarında her bir filtreleme işlemi için: $O(n^3)$ adet karşılaştırma yaparak uygun adayların bulunması gerekmektedir. Bu filtreleme işleminin sorgu çizgesini oluşturan k kazayağın her biri için yapılması gerekmektedir. Bu durumda sadece filtreleme işlemi için $O(k * n^3)$ işlem yapmak gerekmektedir.

Filtreleme maliyetini düşürmek için ortalama erişim süresi $O(1)$ olan karma tabloları indeks yapısı olarak seçilmiştir. Bilindiği üzere karma tabloları ilişkili bir anahtar ile verileri saklamaktadır ve daha sonra bu anahtar ile ilgili veriye erişim sağlamaktadır.



Şekil 3.9: Örnek hedef çizgeye (G) ait kazayağılar ve bunların tutulduğu veri yapısı.

Şekil 3.9'de örnek hedef çizgenin tüm kazayağları ve bu kazayağların tutulduğu veri yapısı gösterilmektedir. Kazayağların hiyerarşik bir karma tabloda tutulmaktadır. Bu karma tablosunun birinci seviyesinde kazayağlarının tepe düğümlerinin ait olduğu parça anahtar olarak kullanılmaktadır. Tepe düğümün ait

olduğu parça bilgisi, sol ve sağ kelimelerinin İngilizce karşılığı olan ‘Left’ ve ‘Right’ kelimelerinin ilk harfleri olacak şekilde temsil edilmektedir. Birinci seviye karma tablosu ikinci seviye karma tablolarından oluşmaktadır. İkinci seviye karma tabloları, üçüncü seviye karma tablolarından oluşmaktadır ve kazayağlarının sol bacak düğümlerinin dereceleri anahtar olarak kullanılmaktadır. Üçüncü seviye karma tabloları dördüncü seviye karma tablolarını saklamaktadır ve anahtar olarak kazayağlarının tepe düğüm derecesi kullanılmaktadır. Dördüncü seviye karma tabloları hiyerarşik anahtarlamaya uygun kazayağlarının saklandığı listeleri, kazayağlarının sağ bacak düğüm dereceleri anahtar olacak şekilde saklamaktadır.

Geliştirilen algoritma filtreleme yaparken kazayağlarının tepe birçok diğer algoritma gibi düğümlerin derecelerinden faydalanmaktadır: Aynı düğüm derece dizilimine sahip olan ve tepe noktası aynı parçada olan kazayağları, aynı anahtar ile karma tablosunda saklanmaktadır. Filtreleme işlemi yapılırken bu bilgi filtreleme işlemini oldukça hızlandırmaktadır.

3.2.2. İndeksleme

Algoritma iki parçalı hedef çizgeyi indekslerken iki uzunluktaki tüm yolları bir kazayağı olarak hesaba katmaktadır. Kazayağlarını oluşturmak için komşuluklardan faydalanılmaktadır: Bu bölüm sadece indekslemeyi içersin, diğer kısımları üste taşı. Her bir düğüm, kendi komşuluk kümesindeki düğümlerin ikili altkümelerine tepe noktası olarak eklenerek tüm kazayağları oluşturulur.

Örnek bir indeksleme Şekil 3.9’da gösterilmektedir. Oluşturulan ilk kazayağın (a) tepe noktası sağ parçada olduğu için ilk anahtar değeri ‘R’ olmaktadır. Sol bacak düğüm derecesi 2, tepe düğüm derecesi 3 ve sağ bacak düğüm derecesi 2 olduğu için karma tablosunda ilk seviye anahtarı R, ikinci seviye anahtarı 2, üçüncü seviye anahtarı 3 ve dördüncü seviye anahtarı iki olan listede saklanmaktadır.

3.2.3. Filtreleme ve Yeniden Oluşturma

Hedef çizgeyi oluşturan kazayağları oluşturulup Şekil 3.9’da gösterildiği gibi bir karma tablosuna doldurulduktan sonra algoritma herhangi bir sorgu çizgesini cevaplamaya hazır hale gelmektedir. Bir sorgu çizgesi ile eşbiçimli olan alt-çizgeleri

bulmak için: Sorgu çizgesi, Şekil 3.8'nın (2) ile gösterilen bölümünde olduğu gibi kendini oluşturan kazayağlarına parçalanır ve her bir kazayağı için uygun adaylar, indeks verileri arasından filtrelenerek seçilir. Adaylar seçildikten sonra, sorgu çizgesini en az maliyetle eksiksiz geri oluşturabilecek yeniden oluşturma listesi (YOL) belirlenir. Son aşamada; Kazayağların aday kümeleri girişim işleminden geçirilir (Tanım 17) ve sorgu çizgesi tekrar elde edilir.

Tanım 17 (Girişim (İng. Join) İşlemi): R ve S olmak üzere iki ilişki verilmiş olsun. R ve S ilişkilerinin çok-öğeli elemanlarının, ortak özelliklere sahip öğeleri aynı değere sahip olacak şekilde birleştirilmesi ile elde edilen kümenin bulunması işlemidir. R ⋈ S olarak gösterilmektedir.

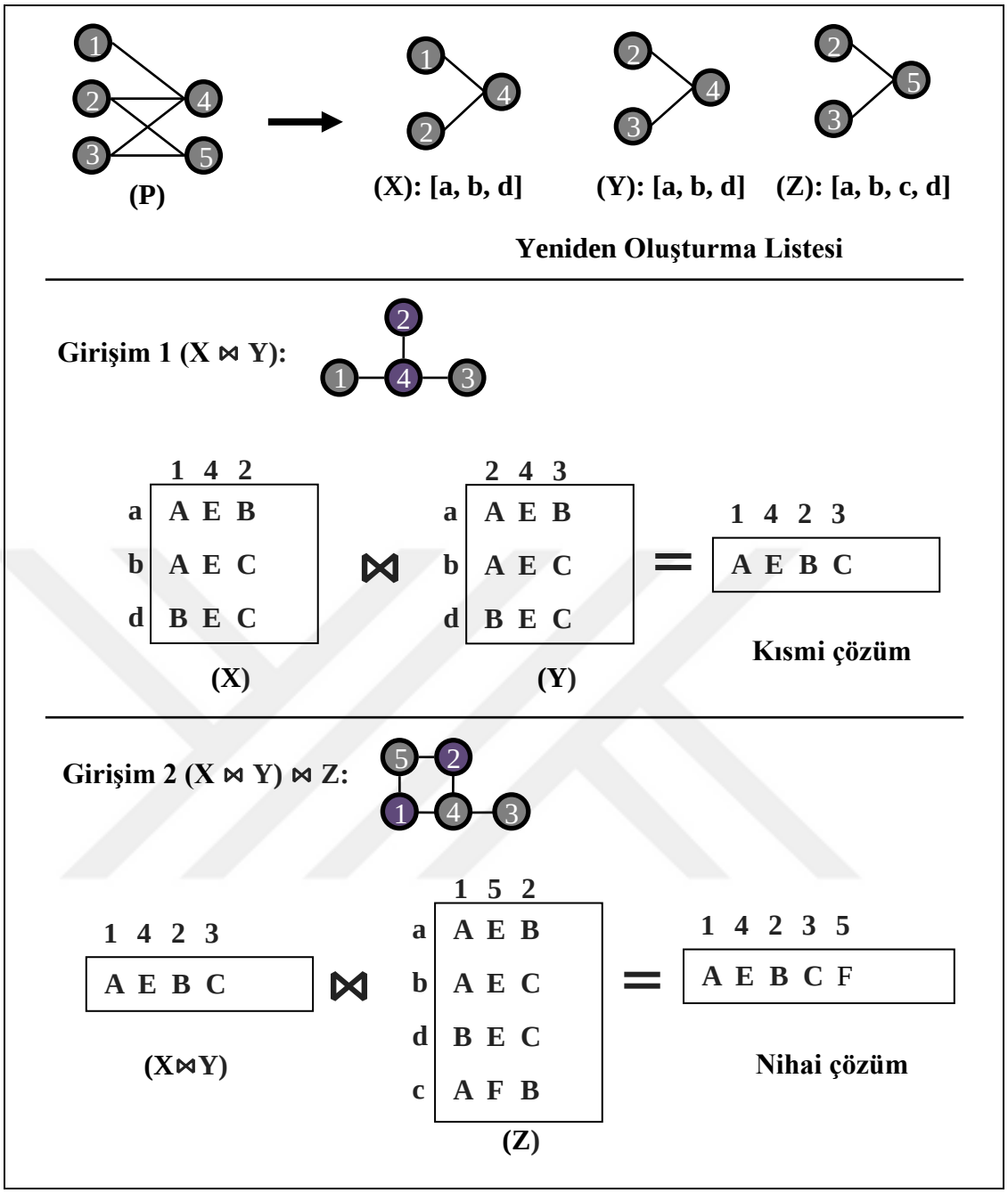
Sorgu çizgesinin kazayağlarına aday seçimi sırasında öncelikle tepe düğümü farklı parçada olan hedef çizge kazayağları filtrelenir. Daha sonra, birinci seviye anahtar değeri sol bacak düğümünün derecesinden düşük olan kazayağları filtrelenir. Birinci seviye anahtar değeri, sol bacak düğümünün derecesine eşit olmak zorunda değildir, sol bacak değeri en az sorgu kazayağın sol bacak değerine eşit olan adaylar yeterli şartı sağlamaktadır. Bu derece kontrol işlemi tepe ve sağ bacağa da sırasıyla uygulanır ve uygun adaylar böylece seçilmiş olur.

Adaylar seçildikten sonra tekrar oluşturma işlemi başlar. Sorgu çizgesinin yeniden oluşturma işlemine hangi kazayağından başlandığına göre; yapılması gereken girişim işlemi sayısı değişmektedir. Sorgu çizgesinin en az sayıda girişim işlemiyle yeniden oluşturulması için; hangi kazayağların kullanıldığı ve bu kazayağların ne sırayla girişim işlemine tabi tutulduğu Bölüm 3.2.4'te detaylı olarak açıklanmaktadır.

```
# YenidenOluşturma (YOL, sorgu_çizgesi)
1  nihai_sonuçlar = [ ]
2  kısmi_çözüm_şablonu = sorgu_çizgesinin her bir düğümünü anahtar değeri
   olarak tutan karma tablosu
3  YOL daki ilk kazayağının her bir adayı k için:
4      kısmi_çözüm = Kısmi_çözüm_şablonundan bir kopya oluştur
5      kazayağının her bir düğümü v için:
6          kısmi_çözüm[v] = k[v]
7      kısmi_çözümü nihai_sonuçlara ekle
8  YOLdan ilk kazayağını sil
9  YOLdaki her bir kazayağı k için:
10     nihai_sonuçlar = nihai_sonuçlar ⋈ k kazayağının adayları
11 nihai_sonuçları döndür
```

Şekil 3.10: Yeniden oluşturma algoritması sözde kodu.

Şekil 3.10 da gösterilen sözde kodda da görüldüğü üzere, YOL elde edildikten sonra, bu listenin ilk kazayağını ele alınarak, bu kazayağın her bir adayı için kısmi bir çözüm üretilmektedir. Daha sonra bu kısmi çözümlerin her biri yeniden oluşturma listesindeki ikinci kazayağın adayları ile girişim işlemine tabi tutulur ve başarılı girişimlerin her biri için yeni bir kısmi çözüm üretilir. Kısmi çözüm oluşturma işlemi, yeniden oluşturma listesindeki tüm kazayağları girişim işlemine tabi tutulana kadar devam eder. Son kazayağın adaylarının da kısmi tüm çözümler ile girişime tabi tutulmasından elde edilen çözümler, nihai çözüm kümesidir.



Şekil 3.11: Hedef çizge (Şekil 3.9) içerisindeki aday kazayağları ile girişim ve yeniden oluşturma.

Şekil 3.11'de sorgu çizgesinin YOL'daki kazayağların girişim örnekleri verilmektedir. Hedef çizge olarak 3.9'da gösterilen G çizgesi kullanılmaktadır. Yeniden oluşturma listesindeki kazayağlarına uygun adaylar bu G çizgesinden seçildikten sonra **X.** ve **Y** kazayağlarının girişimi ile yeniden oluşturma başlamaktadır. **X** kazayağının ilk adayı 'a' ile **Y** kazayağının ilk uygun adayı 'b' arasında başarılı bir girişim mümkün değildir: **X** ve **Y** kazayağları 2 ve 4 düğümleri üzerinden kesişmektedir fakat a ile b aday kazayağlarının bu düğümlerdeki değerleri

aynı değildir. **X** kazayağının tüm adayları, **Y** kazayağının tüm adayları ile girişim işlemine tabi tutulur. **X** kazayağının d adayı ile **Y** kazayağının a adayı arasında başarılı bir girişim yapılabildiği için bu girişimin sonucu kısmi bir sonuç olarak sonraki aşamaya iletilir. Sonraki aşamada bir önceki aşamadan gelen kısmi çözümler, Z kazayağını ile girişim işlemine tabi tutulur. Kısmi çözümler arasındaki (d ⋈ **a)** girişimi ile c aday kazayağın başarılı girişimi sayesinde bir nihai sonuç elde edilmektedir.

M düğüme sahip bir sorgu çizgesinin yeniden oluşturma işlemi için: Sorgu çizgesini oluşturan tüm kazayağların kullanılması durumunda, gerekli girişim sayısı aşağıdaki formül ile hesaplanabilmektedir:

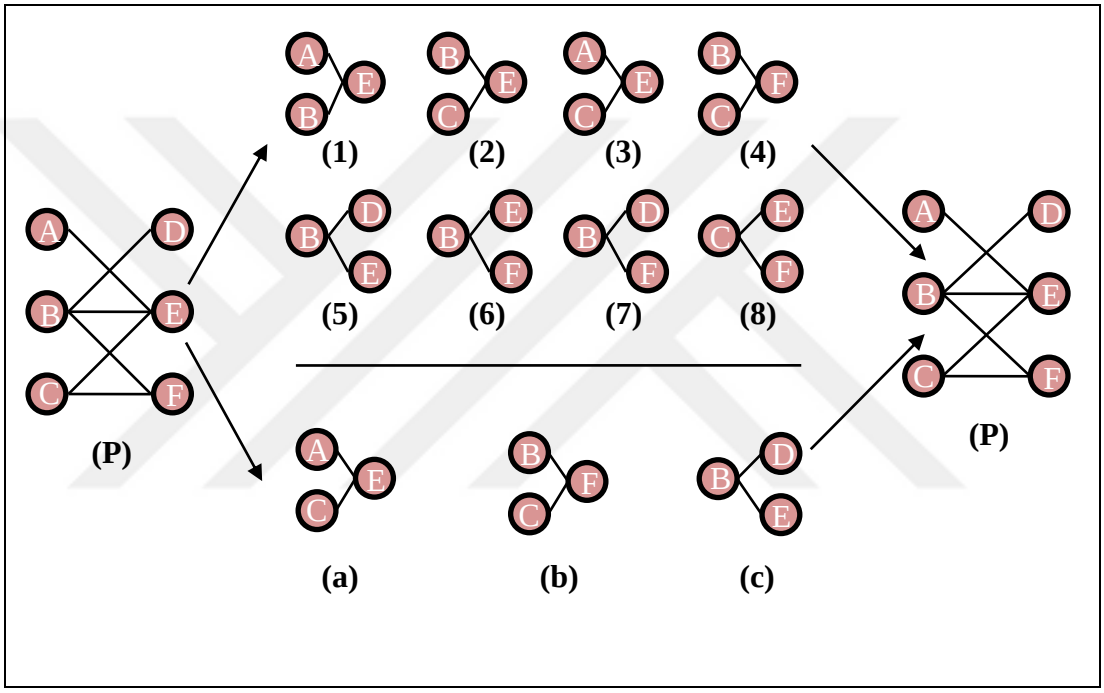
$$|\bowtie| = \prod_{i}^{\binom{M}{3}} |C(T_i)| \tag{3.4}$$

Formülde $C(T_i)$ ile gösterilen ifade, i kazayağın aday kümesini temsil etmektedir. Formülden de görüldüğü üzere altı düğüme sahip bir sorgu çizgesini oluşturan kazayağların ortalama 10 adayı olması $O(10^{20})$ adet girişim işlemi yapılması gerekmektedir. Görece küçük sayılabilecek sorgu ve hedef çizgeler için oldukça maliyetli olan girişim işleminin bu denli fazla olması, algoritmayı kullanılamaz hale getirmektedir. Bu sorunun üstesinden gelmek için sorgu çizgesini oluşturan kazayağları incelendiğinde; çizgenin tüm düğüm ve ayrıtlarının birkaç kazayağı kullanılarak elde edilebildiği, kalan kazayağların sonuçları elde etmeye hiçbir katkıda bulunmadığı görülmüştür. Şekil 3.11'de gösterilen sorgu çizge, altı farklı kazayağına sahip olmasına rağmen sadece üç kazayağı ile tüm düğüm, ayrıt ve komşuluklar eksiksiz olarak ifade edilebilmektedir. Bu durum, hangi kazayağların kullanılması halinde minimum sayıda girişim işlemine ihtiyaç duyulacağı sorusunu ortaya çıkarmıştır. Yeniden oluşturma listesinin seçimi olarak adlandırılan probleme geliştirilen çözüm yaklaşımı bölüm 3.2.4'te detaylı olarak açıklanmaktadır.

3.2.4. YOL Seçimi

Sorgu çizgesi parçalanırken tüm iki uzunluklu yollar bir kazayağı olarak kabul edilmektedir. Elde edilebilen tüm kazayağların yeniden oluşturmada kullanılması

oldukça maliyetli ve gereksizdir. Şekil 3.12'da gösterilen sorgu çizgesi, en kötü durumda 8 kazayağından oluşabildiği gibi (üstte) en iyi durumda (altta) sadece üç kazayağı kullanılarak da oluşturulabilmektedir. Sonuçların üretilmesi için gerekli girişim işlemi sayısı kazayağı sayısı ile doğru orantılı olarak artmaktadır. Girişim sayısının artması algoritmanın performansını olumsuz yönde etkilemektedir. Bu durum göz önünde bulundurulduğunda, en az sayıda girişime sebep olacak yeniden oluşturma listesinin bulunması, algoritmanın performansı için hayati önem taşımaktadır.



Şekil 3.12: İki farklı yeniden oluşturma listesi.

Bu çalışmada en az sayıda girişime ihtiyaç duyan yeniden oluşturma listesini bulmak için iki farklı yöntem önerilmektedir. Geliştirilen ilk yöntem açgözlü yaklaşıma sahiptir ve sözde kodu Şekil 3.13'te gösterilmektedir. Algoritma en az adaya sahip kazayağını seçerek listeyi oluşturmaya başlar. Sonraki her adımda, kalan kazayağları arasından en çok ziyaret edilmemiş ayrıta sahip kazayağları bulunur. Bu kazayağları içerisinden en az adaya sahip olanı seçilerek yeniden oluşturma listesine eklenir. Bu işlem; ziyaret edilen ayrıt sayısı, sorgu çizgesinin ayrıt sayısına eşit olana kadar devam eder. Kısmi liste tüm ayrıtları kapsadığında geçerli bir yeniden oluşturma listesi elde edilmiş olur.

```
# AçgözlüYOLOluşturma (sorgu_kazayağları, sorgu_çizgesi_ayrıtları)
1  YOL = [ ]
2  kapsanan_ayrıtlar = [ ]
3  Tüm sorgu_çizgesi_ayrıtları kapsanan_ayrıtlar listesine eklenene kadar:
4      mevcut_adımdaki_en_iyi_kazayağı = Boş
5      maksimum_kapsanmayan_ayrıt_sayısı = 1
6      sorgu_kazayağları listesindeki her bir kazayağı k için:
7          Eğer k YOL da değilse:
8              Eğer k kazayağının kapsanan_ayrıtlar listesinde olmayan ayrıt
               sayısı maksimum_kapsanmayan_ayrıt_sayısına eşit ise:
9                  Eğer mevcut_adımdaki_en_iyi_kazayağı Boş ise:
10                     mevcut_adımdaki_en_iyi_kazayağı = k
11                 Eğer k kazayağının aday sayısı mevcut adımdaki en iyi
               kazayağının aday sayısından az ise:
12                     mevcut_adımdaki_en_iyi_kazayağı = k
13             Eğer k kazayağının kapsanan_ayrıtlar listesinde olmayan ayrıt
               sayısı maksimum_kapsanmayan_ayrıt_sayısından büyük ise:
14                 mevcut_adımdaki_en_iyi_kazayağı = k
15                 geçici_maksimum_kapsanmayan_ayrıt_sayısı = k kazayağının
                   kapsanan_ayrıtlar listesinde olmayan ayrıt sayısı
16     mevcut_adımdaki_en_iyi_kazayağını YOL'a ekle
17 YOL'u döndür
```

Şekil 3.13: Açgözlü YOL oluşturma algoritması sözde kodu.

Üzerinde çalışılan hedef ve sorgu çizgelerinin bağlı çizgeler olduğu kabul edilmektedir. Bağlı çizgelerde tüm ayrıtlar ziyaret edildiğinde; çizgenin tüm düğümleri de ziyaret edilmiş olmaktadır. Bu durumda üretilen yeniden oluşturma listesinin, sorgu çizgesini eksiksiz oluşturabileceği garanti altına alınmış olmaktadır.

Açgözlü algoritmanın performansı yeniden oluşturma listesindeki kazayağı sayısı ile doğrudan ilişkili olduğu için algoritmanın çalışma süresi analizi kazayağı sayısı üzerinden yapılabilir. Algoritmanın en kötü durumda üreteceği listenin uzunluğunu hesaplamak için: En kötü durumda tüm düğümleri kapsayan kazayağları

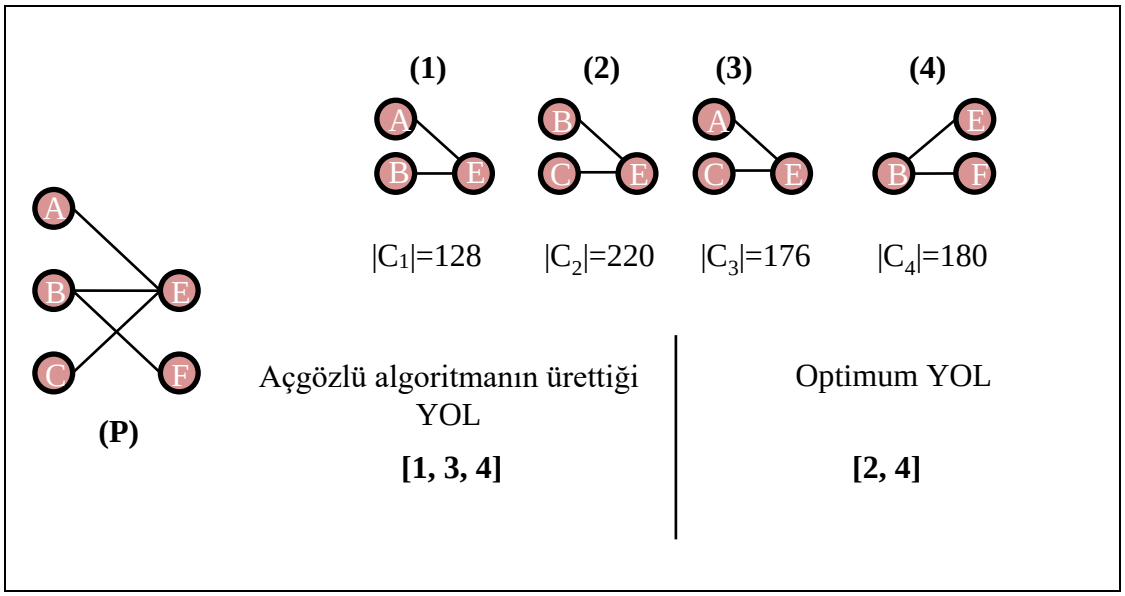
ile tüm ayrıtları kapsayan kazayağların birleşim kümesi bulunmalıdır. Bir kazayağı en iyi durumda iki, en kötü durumda sıfır ziyaret edilmemiş ayrıtı kapsayabilmektedir.

Bir düğümü içeren kazayağlarından en fazla bir tanesi, bir ziyaret edilmemiş ayrıtı kapsamak için yeniden oluşturma listesine eklenmiş olabilir. Aksi durumda, eğer bir düğüm üzerinde $2k$ +1 adet tekil ayrıt kapsayan kazayağı varsa: Bu kazayağların k tanesi tek bir iki ayrıt kapsayan kazayağına dönüştürülebilir ve yine 1 adet tekil kapsayan kazayağı kalır. En kötü durumda her bir düğümün tekil kapsayan ayrıta sahip olması mümkündür ve böylece $(|U| + |V|)$ adet ayrıt, sadece bir ziyaret edilmemiş ayrıt kapsayan kazayağları tarafından kapsanmaktadır. Geriye kalan $|E| - (|U| + |V|)$ ayrıt ise iki ziyaret edilmemiş ayrıta sahip kazayağları tarafından kapsanmaktadır. Bu durumda her iki ayrıt bir kazayağı tarafından kapsandığı için: $\frac{|E|-(|U|+|V|)}{2}$ adet kazayağına ihtiyaç duyulmaktadır. Böylece Her iki durumu karşılayan toplam kazayağı sayısı:

$$|YOL| = O\left((|U| + |V|) + \left(\frac{|E| - (|U| + |V|)}{2}\right)\right)$$
$$|YOL| = O\left(\frac{|E| + (|U| + |V|)}{2}\right) \quad (3.5)$$

En kötü durum analizi göstermektedir ki açgözlü algoritma ile gerekli kazayağı sayısını $O((|V|^2 * |U|) + (|U|^2 * |V|))$ 'den $O(|E| + |U| + |V|)$'ye düşürmektedir.

Açgözlü yaklaşım gerekli kazayağı sayısını düşürse de bazı durumlarda optimal YOL'u üretememektedir. Şekil 3.14'da gösterilen sorgu çizgesi incelendiğinde; mevcut algoritma, her adımda, en çok ziyaret edilmemiş ayrıta sahip kazayağları içerisinden en az adaya sahip kazayağını seçtiği için ilk olarak (1) numaralı kazayağını seçer ve listeye ekler. Daha sonraki adımda tüm kazayağları sadece bir ziyaret edilmemiş ayrıta sahip olduğu için kalan kazayağları içerisinden en az adaya sahip olan (3) numaralı kazayağı seçilerek listeye eklenmektedir. Son olarak kalan iki kazayağından (2) numaralı kazayağı çözüme katkı sağlamadığı için (4) numaralı kazayağı seçilerek tüm çizge kapsanmaktadır.



Şekil 3.14: En az girişime sahip yeniden oluşturma listesi (sağda) ve açgözlü algoritmanın ürettiği yeniden oluşturma listesi (solda).

Elde edilen yeniden oluşturma listesi her ne kadar dört kazayağın hepsini kullanmaktan iyi olsa da çizge tekrar incelendiğinde sadece (3) ve (4) numaralı kazayağları ile çizgenin eksiksiz biçimde yeniden oluşturulabildiği gözlemlenmektedir. Açgözlü algoritmanın ürettiği çözüm en kötü durumda 128 x 176 x 180 girişime ihtiyaç duyarken optimum çözüm sadece 176 x 180 adet girişim ile sonuçları üretebilmektedir. Hesaplamalar gösteriyor ki optimum liste, açgözlü algoritmanın ürettiği listeden, en kötü durumda, 128 kat daha iyi performansa sahiptir.

Yapılan tez çalışmasında kaba-kuvvet yaklaşımına sahip bir başka yöntem daha kullanılmaktadır. Bu yöntem, çizgeyi kapsayan tüm yeniden oluşturma listelerini oluşturmaktadır. Oluşturulan tüm listelerin ihtiyaç duyduğu girişim sayıları hesaplanarak en az sayıda girişime ihtiyaç duyan liste optimum liste olarak belirlenir. Bu algoritma her zaman en az sayıda girişime ihtiyaç duyan yeniden oluşturma listesini bulmayı garanti eder fakat doğası gereği oldukça maliyetlidir. Bir sorgu çizgesini oluşturan tüm kazayağların sayısı *k* olsun: Algoritma, olası tüm kazayağları içerek kümenin tüm alt kümelerini ürettiği için 2^k adet liste oluşturmak zorundadır. Daha önceden belirtildiği üzere dengeli bir iki parçalı çizgede $\binom{|U|}{3}$ adet kazayağı bulunmaktadır. Bu durumda optimum YOL'un bulunması için gerekli işlem sayısı aşağıdaki formülle hesaplanabilir:

$$2^{\binom{|U|}{3}} = 2^{\frac{|U|*(|U|-1)*(|U|-2)}{3*2}} = O\left(2^{|U|^3}\right) \quad (3.6)$$

Denklemden de anlaşıldığı üzere oldukça maliyetli olan bu algoritma, ortalama bir sistemde, en fazla 8 düğüme sahip sorgu alt-çizgeleri için kullanılması makuldür. Tüm düğüm ve ayrıtları kapsayan herhangi bir yeniden oluşturma kümesi aynı sayıda sonuç üretecektir. Bu kümelerin içerdiği kazayağların girişim sırası da oluşacak sonuçlara etki etmeyecektir. Diğer yandan yeniden oluşturma listesindeki kazayağların sırası değiştiğinde, algoritmanın performansında farklılıklara sebep olmaktadır. Bu durumun sebebi ara adımlarda oluşturulan kısmi çözümlerin sayısıdır. Her bir girişim adımında bir sonraki kazayağı ile girişimi sağlanmak üzere kısmi sonuçlar üretilmektedir. Bu kısmi sonuçların birçoğu daha sonraki adımlarda uygun bir girişim sağlayamayacak ve elenecektir. Bu tür sonuç üretmesi imkânsız ara çözümler ne kadar erken elenebilirse, algoritmanın sonraki adımlarda yapması gereken girişim sayısı o kadar azalacaktır. Girişim sayısının azalması ise algoritmanın performansını arttıracaktır.

Bu çalışmada en iyi girişim sırasının belirlenmesi için iki parametre göz önünde bulundurulur: yeniden oluşturma listesindeki ardışık kazayağların kesişimi ve bu kazayağların aday sayıları. Bu parametrelerin, hangi ikili parametre değerinin en iyi olduğu performans açısından karşılaştırmalı test edilmiştir. Testler, en fazla sayıda adaya sahip kazayağından başlanır ve her bir adımda mevcut çözümle kesişen en yüksek adaya sahip kazayağı ile devam edilirse, algoritmanın en az sayıda girişim ile sonuçları elde edebildiğini göstermektedir. Bu durumun sebepleri incelendiğinde, bağlı kazayağların girişimi sırasında kısmi çözümlerin birçoğunun uygunsuz girişim sebebiyle elendiği görülmektedir. Aday sayısı yüksek olan kazayağlarına öncelik verildiğinde, uygunsuz girişim sebebiyle elenen kısmi çözüm sayısı maksimum seviyeye çıktığı ve bu sayede performansın arttığı görülmektedir.

4. İKİ PARÇALI ÇİZGELERDE YAKLAŞIK EŞLEME

Geliştirilen yaklaşık eşleme yöntemi, Bölüm 3.2'de anlatılan indeks tabanlı algoritmanın, eşleme şartlarının esnetilmesine dayanmaktadır. Sınır tabanlı yaklaşıma sahip bu algoritma, ayrıtların eksikliğine tolerans göstererek yaklaşık eşleme yapmaya imkân tanımaktadır. Hedef çizge içerisinde, sorgu çizgesi ile aynı sayıda düğüme ve komşuluk ilişkilerine sahip fakat belirli bir sınır değerinden daha az sayıda ayrıtı eksik olan tüm alt-çizgeler algoritma tarafından bulunmaktadır.

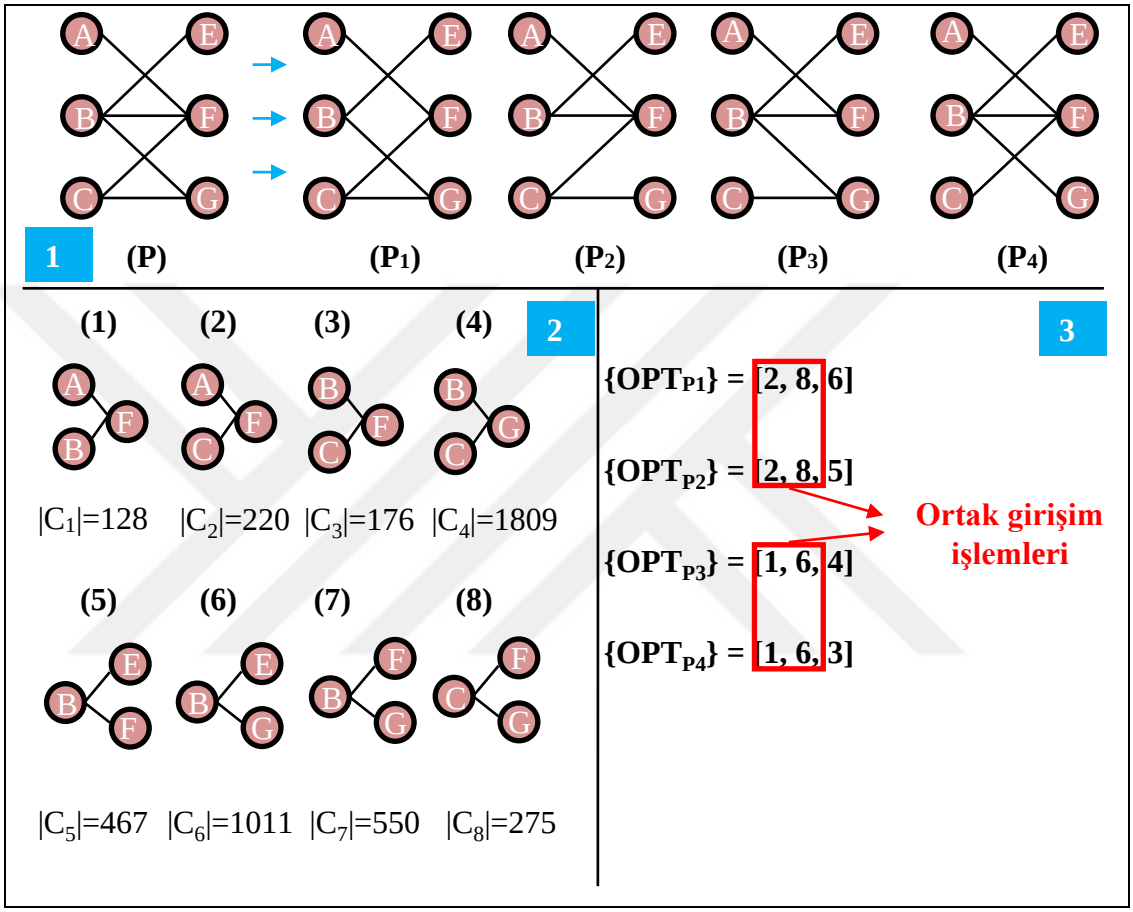
Sadece ayrıtların eksikliği üzerinden gerçekleştirilen türde bir rahatlatma, hedef çizgenin düğümleri arasında eksik bağlantı (ayrıt) olduğu durumlarda oldukça işe yaramaktadır. Örnek olarak; insan proteinleri ile patojen proteinleri arasındaki etkileşim ağlarında henüz keşfedilmemiş ya da deneyler sırasında eksik saptanmış etkileşimler bulunabilmektedir. Bu tür bir ağın modellendiği çizge üzerinde birebir eşleme işlemi dolayısıyla yetersiz kalmaktadır. Böyle bir ağ içerisinde yeni etkileşimler tahmin etmek ya da eksik etkileşimlere rağmen başarılı bir eşleme yapmak için, bir yaklaşık eşleme algoritmasına ihtiyaç duyulmaktadır. Bu amaçla, geliştirilen indeks tabanlı algoritmayı temel alan bir yaklaşık eşleme yöntemi geliştirilmiştir.

Yöntemin temel çalışma prensibi, k adet ayrıtın esnetilmesine izin verildiğinde, sorgu çizgesi ile arasında çizge benzetim uzaklığı k'yı geçmeyen tüm çizgeler için birebir eşleme yapmak üzerinedir. Böyle bir yaklaşım doğası gereği çok maliyetlidir. Sınır değeri k olarak belirlendiği durumda, eşleme yapılacak sorgu çizgelerinin ($\{P\}$) sayısı aşağıdaki denklem ile hesaplanabilmektedir:

$$|\{P\}| = \binom{|E|}{k} + \binom{|E|}{k-1} + \binom{|E|}{k-2} + \cdots + \binom{|E|}{1} \tag{4.1}$$

Adet çizgenin birebir eşlenmesi gerekmektedir. Böyle kaba-kuvvet bir yaklaşımla problemi çözmek oldukça verimsiz olduğundan orta ölçekli çizgelerde bile problemin çözümü imkânsız hale gelmektedir. Diğer yandan eşleme yapılacak çizgelerin birbirinden en fazla k adet ayrıt farklı olması, bu çizgeler birebir eşlenirken birçok ortak işleme tabi tutulması gerektiği fikrini doğurmaktadır.

Şekil 4.1'de sorgu alt-çizgesinin bir ayrıtının eksikliği ile elde edilebilen tüm çizgeler ve bu çizgeleri oluşturmak için gerekli olan optimum yeniden oluşturma listeleri görülmektedir. Listeler incelendiğinde, öngörüldüğü üzere, birçok ortak kazayağı girişimi olduğu gözlemlenmektedir çünkü listeler aynı seçim stratejisi ile oluşturulmaktadır.



Şekil 4.1: P sorgu çizgesinin bir ayrıt eksik alt-çizgeleri (1), Çizgeyi oluşturan kazayağları (2) ve bir ayrıtı eksik alt-çizgelerin optimal YOL'ları (3).

Bu tez çalışması kapsamında; ortak kazayağlarından faydalanarak yaklaşık eşleme işlemi yapan iki farklı yöntem geliştirilmiştir. İlk yöntemin temel mantığı, en çok beliren kazayağların öncelikle girişimini gerçekleştirerek ortak kazayağlarından maksimum oranda faydalanmaktır. Algoritma bu işlem için; yeniden oluşturma listeleri içerisinde en çok beliren kazayağı ile bu kazayağını içeren listelerdeki en çok beliren ikinci kazayağını girişim işlemine tabi tutar. Daha sonra tüm listelerde en çok beliren ikinci kazayağı ele alınır ve aynı işlem tekrarlanır. Bu girişim işlemleri, girişime tabi tutulan kazayağı kimlikleri anahtar olacak şekilde bir karma tablosunda

saklanır. Bu girişim işlemleri, listelerin en az yarısında beliren bir kazayağı kalmayana kadar devam eder. Daha sonra bu ikili girişimler üçlü girişimlere genişletilmek için listeler tekrar taranır. İkili girişimi oluşturan kazayağların ortak bulunduğu listelerde en çok beliren kazayağı ile girişim sağlanarak üçlü alt çözümler elde edilir. Yine bu işlemde ikili yapının kazayağlarının bulunduğu listelerin en az yarısında bulunan bir kazayağı kalmayana kadar devam eder. Bu genişletme işlemi, herhangi bir adımda alt çözümü oluşturan kazayağların bulunduğu listelerin en az yarısında beliren bir kazayağı kalmayana kadar devam eder.

Bir listeye ait sonuçlar elde edilirken; öncelikle listenin en uzun alt çözümlerden birini içerip içermediğine bakılır. Eğer en uzun alt çözümlerin hiçbirini barındırmıyorsa; en uzun çözümlerden bir kazayağı eksik çözümler için aynı işlem tekrarlanır. Bu sayede yeniden oluşturma listelerinde bulunan, daha önceden hesaplanmış en uzun girişimlerin kullanılması sağlanır. Önceden hesaplanmış, ortak girişimler ile kısmi çözümler oluşturulduktan sonra, algoritma kalan kazayağların da girişimlerini Bölüm 3.2.3'de anlatıldığı gibi tamamlayarak nihai sonuçları elde eder. Bu işlem tüm yeniden oluşturma listelerine uygulanır ve elde edilen sonuçlar bir araya getirilerek yaklaşık eşleme sonuçları elde edilir.

Bahsi geçen yaklaşım birçok tekrar eden kazayağı girişim işleminin önüne geçse de her bir k ayrıtı eksik varyasyon için optimum yeniden oluşturma listesinin oluşturulması ve bu listeler içerisinden ortak ikili ve üçlülerin belirlenmesi maliyetli işlemlerdir. Diğer yandan, çoğu durumda listeler arasında ortak kazayağları bulunsa da yeniden oluşturma listelerinin yaklaşık olarak yarısı birbirinden farklılık göstermektedir. Son olarak bu listelerdeki girişim sıraları daha önceden bahsedildiği üzere oldukça önemlidir ve küme tabanlı bir yaklaşım bu faydayı ortadan kısmen de olsa kaldırmaktadır. Bu problemlerin üstesinden gelen ve dolayısıyla yapılması gereken girişim sayısını daha etkili bir şekilde azaltan ikinci bir yöntem daha tez çalışması kapsamında geliştirilmiştir.

İkinci yönteme ait sözde kod Şekil 4.2'de gösterilmektedir. Bu yöntemde sorgu çizgesinden k adet ayrıt çıkarıldığında, bu çizgenin yeniden oluşturma listesinde maksimum $2k$ kazayağın bu işlemden etkileneceği gerçeğine dayanmaktadır. Yeniden oluşturma listelerinde bir ayrıt maksimum iki farklı kazayağı tarafından kullanılabilmektedir: Üç kazayağı aynı ayrıtı kullanıyorsa eğer bu kazayağlarından ikisi tek bir kazayağına dönüştürülebilir ve böylece yine maksimum iki kazayağı tarafından kapsanmış olur.

```
# YaklaşıkEşlemeYOLOluşturma (y_sorgular, opt_yol, sorgu_kazayağları)
1  yaklaşık_sorgu_YOLları = [ ]
2  yaklaşık_sorgular listesindeki her bir yaklaşık sorgu p için:
3      geçici_YOL = optimal_yol
4      silinen_kazayağları = [ ]
5      geçici_YOLdaki her bir kazayağı k için:
6          Eğer kazayağının ayrıtları p kısmi sorgusunda yok ise:
7              k yı geçici_YOLdan sil
8              k yı silinen_kazayağları listesine ekle
       kısmi_kapsanan_ayrıtlar = geçici_YOLdaki kazayağlarının ayrıtları
9      p yaklaşık sorgusunda olup kısmi_kapsanan_ayrıtlarda bulunmayan her bir
10     e ayrıtı için:
11         minimum_aday_sayısı = Sonsuz büyüklükte tam sayı
           kısmi_en_iyi_aday = Boş
12         sorgu_kazayağları_listesi içerisindeki her bir kazayı k için:
13             Eğer k geçici_YOLda ve silinen_kazayağları listesinde değilse:
                   Eğer k kazayağının ayrıtlarından biri e ise ve k nın aday sayısı
14                 minimum_aday_sayısından az ise:
15                     kısmi_en_iyi_aday = k
               kısmi_en_iyi_adayı geçici_YOLa ekle
16         geçici_YOLu yaklaşık_sorgu_YOLlarına ekle
17     yaklaşık_sorgu_YOLlarını döndür.
```

Şekil 4.2: Yeniden oluşturma listelerinin oluşturulması sözde kodu.

Algoritma bu durumdan faydalanmak için sorgu çizgesi için optimal yeniden oluşturma listesini bulmaktadır. *k* ayrıtı eksik olan bir alt çizge için yeniden oluşturma listesi belirlenirken optimum listeden faydalanılır: Optimum listeden, alt-çizge varyasyonunu elde etmek için çıkarılan ayrıtları içeren kazayağları çıkarılır. Bu çıkarma işlemi sebebiyle bazı ayrıtlar, çıkarılan kazayağların diğer bacaklarında bulunmaları sebebiyle istemsiz olarak yeniden oluşturma listesinin kapsadığı ayrıtlar kümesinin dışında kalırlar. Bu ayrıtları tekrar yeniden oluşturmaya dahil etmek için: Alt-çizgeyi elde ederken çıkarılan ayrıtları içermeyen fakat istemsizce çıkarılan

ayrıtları içeren kazayağları arasından bir dizi kazayağı seçilerek yeniden oluşturma listesine eklenir. Bu seçim işlemi indeks tabanlı algoritmada anlatılan kaba-kuvvet yaklaşımı ile gerçekleştirilir ve en az girişime sebep olacak kazayağı dizilimi ile yeniden oluşturma listesi tamamlanır.

Bu çıkarma ve ekleme işlemi tüm *k* ayrıtı eksik alt-çizgenin yeniden oluşturma listesini elde etmek için tekrarlanır. Elde edilen bu listelerin birçoğu ön ek olarak birbirine benzemektedir ve birbirlerinden en fazla 2*k* kazayağı ile ayrılmaktadırlar. Bu motivasyon ile listelerdeki kazayağları birer katar olarak düşünüldüğünde, listeler birbirleri ile birçok ortak öneke sahip olmaktadır. Bu bilgi, ortak girişimlerin etkili biçimde kullanılması adına önem taşımaktadır.

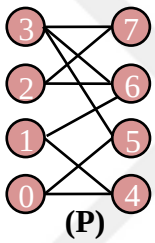
```
#  YaklaşıkEşleme (yaklaşık_sorgu_YOLları)
1  girişim_tablosu = Karma tablo
2  nihai_çözümler = [ ]
3  yaklaşık_sorgu_YOLlarındaki her bir YOL p için:
4      anahtar_kataru = Boş katar
5      p listesindeki her bir kazayağı k için:
6          anahtar_kataru += k kazayağının kimlik etiketi
7      kısmi_çözümler = anahtar_katarının girişim_tablosunda anahtar olarak
       bulunan en uzun ön ekindeki kısmi çözümler
8      Bulunan en uzun önekten sonraki kazayaklarının her bir k kazayağı için:
9          kısmi_çözümler = kısmi_çözümler ⋈ k kazayağının adayları
10     kısmi çözümleri nihai_çözümler listesine ekle
11 nihai_çözümleri döndür.
```

Şekil 4.3: Yaklaşık eşleme algoritması sözde kodu.

Şekil 4.3'te yaklaşık eşleme için yeniden oluşturma algoritmasının sözde kodu gösterilmektedir. Listelerin ortak kazayağı dizilimlerinden faydalanmak için bir karma tablosundan faydalanılmaktadır. Bir alt-çizge için yeniden oluşturma listesi seçilirken bu alt-çizgeye ait kataın, karma tablosunda anahtar olarak bulunan en uzun ön eki aranır. Bu öneke sahip sonuçlar getirilerek her biri için bir kısmi sonuç üretilir. Daha sonra kataın kalan kısmı içerisindeki en uzun önek aranır. Bulunan

sonuçlar ile kısmi sonuçlar arasında girişim işlemi uygulanır. Bu önek bulma ve kısmi sonuçların genişletilmesi işlemi tüm katar kapsanana kadar devam eder. Her girişim işlemi sonunda sonuçlar girişimi oluşturan kazayağların dizilimi anahtar olacak şekilde karma tablosuna kaydedilir.

Şekil 4.4'de, geliştirilen her iki algoritmanın P sorgu çizgesinin bir ayrıt eksik alt-çizgeleri için ürettiği yeniden oluşturma listeleri görülmektedir. Listeler incelendiğinde birçoğunun ortak kazayağlarına, benzer sırada girişim uygulaması gerektiği görülmektedir. Bu durum girişim sayısının düşürülmesinde oldukça faydalıdır: Hiçbir iyileştirme olmaksızın tüm alt-çizgeleri elde etmek için 40 kazayağı girişimi işlemine gerek duyulurken; küme tabanlı algoritma bu işlemi 21, optimum yeniden oluşturma listesine ekleme çıkarma yaparak yeniden oluşturma listeleri üreten algoritma sadece 17 girişimle sonuçları elde edilebilmektedir.



(P)

E = (0-4): **a** (0-5): **b** (2-6): **e** (2-7): **f** (3-7): **i**
(1-4): **c** (1-6): **d** (3-5): **g** (3-6): **h**

Kazayağı Listesi

0 – [0, 4, 1]
1 – [0, 5, 3]
2 – [1, 6, 2]
3 – [1, 6, 3]
4 – [2, 6, 3]
5 – [2, 7, 3]
6 – [0, 4, 1]
7 – [0, 5, 3]
8 – [1, 6, 2]
9 – [1, 6, 3]
10 – [2, 6, 3]
11 – [2, 7, 3]

Optimal YOL'lar	Dizilim Öncelikli YOL'lar
	Optimum YOL: **[0, 1, 3, 4, 5]**
E – a : [1, 4, 5, 7]	**E – a : [x, 1, 3, 4, 5], 7**
E – b : [0, 2, 5, 8]	**E – b : [0, x, 3, 4, 5], 8**
E – c : [1, 4, 5, 7]	**E – c : [x, 1, 3, 4, 5], 6**
E – d : [0, 1, 4, 5]	**E – d : [0, 1, x, 4, 5]**
E – e : [0, 1, 3, 5]	**E – e : [0, 1, x, 3, 5]**
E – f : [0, 1, 3, 10]	**E – f : [0, 1, 3, 4, x], 11**
E – g : [0, 2, 5, 10]	**E – g : [0, x, 3, 4, 5], 6**
E – h : [0, 1, 2, 5]	**E – h : [0, 1, x, x, 5], 2**
E – i : [0, 1, 3, 10]	**E – i : [0, 1, 3, 4, x], 10**

Şekil 4.4: Geliştirilen her iki algoritmanın, P sorgu çizgesinin bir ayrıt eksik alt çizgeleri için ürettiği yeniden oluşturma listeleri.

Geliştirilen her iki yöntem de girişim sayısını oldukça düşürmektedir. Diğer taraftan geliştirilen ikinci yöntem sayesinde, ilk yöntemin ihtiyaç duyduğu maliyetli ön işlemlere gerek kalmamaktadır: Her bir alt-çizge için yeniden oluşturma listesi bulmak yerine; sadece bir kez sorgu çizgesi için optimum liste bulunduktan sonra

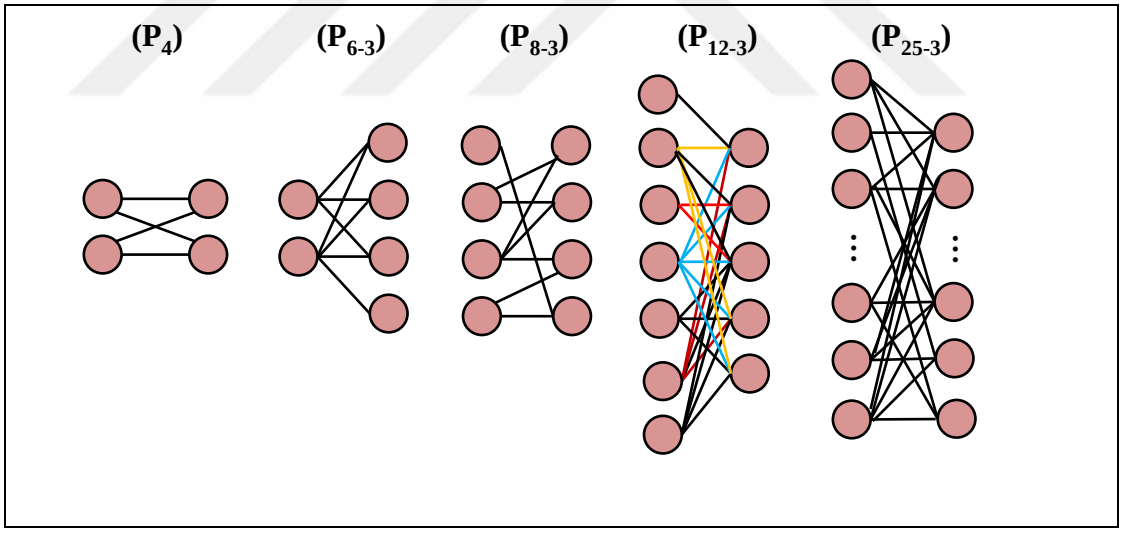
eksik kalan ayrıtları tamamlayacak lokal bir arama gerçekleştirilmektedir. Sıralama tabanlı yöntemin, küme tabanlı yönteme üstün olmasının bir diğer sebebi de listelerin ortak girişimlerinin aranmasına gerek kalmamasıdır: Ön eklerin karma tablosunda indekslenmesi sayesinde benzer girişimlere $O(1)$ sürede erişmek mümkündür.

Tez çalışması kapsamında geliştirilen tüm algoritmalar ve bu algoritmaların kullandığı yöntemler bu bölüm itibari ile tamamen sunulmuştur. Bir sonraki bölümde geliştirilen algoritmalar ile literatürdeki yüksek performanslı rakiplerinin karşılaştırmalı performans testleri bildirilmektedir. Bu testlere ek olarak, algoritmalar geliştirilirken karşılaşılan alt problemlerin çözümüne yönelik yöntemlerin performans karşılaştırmalarına da yer verilmektedir.

5. DENEYSEL VERİ VE DEĞERLENDİRME

Alt-çizge eşbiçimlilik probleminin çözümü için geliştirilen bir algoritmanın değerli olabilmesi için: Algoritmanın hedeflediği problem varyasyonuna, hedeflenen çizge boyutlarında çözüm sunan yüksek performanslı rakiplerine çalışma süresi, sonuç kalitesi veya hafıza tüketimi açısından üstünlük sağlamalıdır. Tez çalışması kapsamında geliştirilen algoritmaların her biri, hedeflediği problem türü için geliştirilen ve kendi ile aynı çözüm yaklaşımına sahip algoritmalar ile performans testlerine tabi tutulmaktadır.

Geliştirilen dal-ve-sınır yaklaşımına sahip birebir eşleme algoritması VF2 [Cordella et al., 2004] ve Turbo^{iso} [Han and Lee, 2013] algoritmaları ile çalışma süreleri üzerinden karşılaştırılmaktadır. İndeks tabanlı algoritma; GADDI [Zhang, Li, and Yang, 2009] algoritması ile karşılaştırılmıştır. Özgün yaklaşık eşleme algoritması, TALE [Tian and Patel, 2008] algoritması ile çalışma süreleri ve çözümün kalitesi üzerinden karşılaştırılmaktadır.



Şekil 5.1: Karşılaştırma testlerinde kullanılan bazı sorgu çizgeleri.

Algoritmalar karşılaştırılırken farklı büyüklükte ve yoğunlukta sorgu çizgeleri kullanılmıştır. Şekil 5.1'de birkaç örneği gösterilen sorgu çizgelerinin tamamı detay bilgileri ile birlikte Tablo 5.1'de bildirilmektedir. Kullanılan sorgu çizgeleri farklı boyut ve yoğunluğa sahiptir. Bu sayede olası tüm sorgu çizgelerini kapsayacak şekilde hazırlanmıştır.

Tablo 5.1: Testlerde Kullanılan Sorgu Çizgeleri.

Çizge Kimliği	Sol Parça D. Sayısı	Sağ Parça D. Sayısı	Ayrıt Sayısı	Ortalama Düğüm Derecesi
P_4	2	2	4	2
$P_{6\text{-}1}$	3	3	6	2
$P_{6\text{-}2}$	3	3	7	2,3
$P_{6\text{-}3}$	2	4	7	2,3
$P_{8\text{-}1}$	4	4	16	4
$P_{8\text{-}2}$	3	5	7	1,75
$P_{8\text{-}3}$	4	4	8	2
$P_{12\text{-}1}$	6	6	27	4,5
$P_{12\text{-}2}$	8	4	13	2,2
$P_{12\text{-}3}$	7	5	24	4
$P_{25\text{-}1}$	12	13	24	1,9
$P_{25\text{-}2}$	8	17	33	2,64
$P_{25\text{-}3}$	13	12	124	10

Yapılan tüm testler doğruluğunun kabulü için 10'ar kez tekrar edilerek elde edilen değerlerin aritmetik ortalaması referans alınmaktadır. Tezin bu bölümünde deneylerin gerçekleştirildiği araçlar, karşılaştırma için kullanılan algoritmalar ve veri setleri tanıtıldıktan sonra geliştirilen algoritmalara ait deneysel testler ve elde edilen sonuçlar alt başlıklar halinde bildirilmektedir.

5.1. Deney Kurulumu

5.1.1. Deney Ortamı

Tez kapsamında gerçekleştirilen testler için kullanılan sistem: 4 çekirdekli i7 970 3.20 Ghz işlemci, 16 GB ana bellek ve 1 TB saklama alanına sahip bir hard disk üzerinde Ubuntu 18.4 işletim sistemini koşmaktadır. Geliştirilen tüm algoritmalar Python 3.6 programlama dili kullanılarak kodlanmıştır. Çizge girdi formatı olarak GraphML [Brandes et al., 2001] çizge formatı kullanılmaktadır.

5.1.2. Hedef Veri Kümeleri

Geliştirilen algoritmalar iki parçalı çizgeleri hedef aldığı için alt-çizge eşbiçimlilik problemine çözüm üreten algoritmaların testlerinde yaygın olarak kullanılan AIDS, NASA ve YEAST [Lee et al., 2012] gibi veri kümeleri kullanılamamaktadır. İhtiyaç duyulan veri kümeleri iki parçalı çizge olarak organize edilebilmelidir. İki parçalı çizge olarak modellenebilen veri kümelerinden insan proteinleri ile patojen proteinleri arasındaki etkileşim ağları ve seçim verileri kümeleri hedef veri kümesi olarak kullanılmıştır. Gerçek veri kümelerinin yanı sıra test edilen algoritmaların hedef aldığı çizgelere uygun olarak üretilen sentetik veri kümeleri de kullanılmıştır. Bu sentetik veri kümeleri Python programlama dilindeki Networkx [Hagberg et al., 2008] kütüphanesi kullanılarak üretilmiştir. Üretilen bu çizgeler yönsüz, ağırlıksız ve bağlı çizgelerdir.

Gerçek veri kümelerinden olan insan proteinleri patojen proteinleri arasındaki etkileşimleri içeren protein etkileşim veri kümesi PHISTO [Durmuş et al., 2013] veri tabanından elde edilmiştir. Bu veri kümesi toplamda 1052 adet küçük ve orta büyüklükte veri kümesi içerdiği için bir çizge veri tabanı olarak organize edilmektedir. Bu küçük ve orta ölçekli çizgeler ortalama 60 düğüm ve 90 ayrıta sahiptir ve her biri, insan ile bir özel bir patojen türü arasındaki protein etkileşim verilerini tutmaktadır. Veri kümesinin özellikleri gereği modellendiği çizge; yönsüz, ağırlıksız ve iki parçalı çizge yapısındadır.

Seçim verileri 2009 yılında online bir sözlük [Leskovec et al., 2010] için gerçekleştirilen yönetici seçimine aittir. Veri kümesi, yaklaşık 7 bin düğüm ve 100 bin ayrıttan oluşan tek bir büyük çizge olarak modellenebilmektedir. Deneylerin üzerinde koşulduğu sistem bu büyüklükte bir çizge için işlem yapamamaktadır bu sebeple bu çizge daraltılarak kullanılmaktadır.

Dal-ve-sınır algoritmasında kullanılan daraltılmış seçim veri kümesi 2200 ayrıt, 520 sol parça düğümü ve 500 sağ parça düğümü içermektedir. Bu algoritmalar test edilirken kullanılan bir diğer veri kümesi Herpesviridae virüsüne ait proteinler ile insan proteinleri arasındaki protein etkileşimlerini içeren veri kümesidir. Bu veri kümesi 140 sol parça düğümü, 730 sağ parça düğümü ve 1155 ayrıttan oluşmaktadır. Dal-ve-sınır algoritmalarını test ederken kullanılan sentetik veri kümesi 280 sol parça düğümü, 300 sağ parça düğümü ve 1000 ayrıttan oluşmaktadır.

İndeks tabanlı algoritmalar ile yapılan testlerde seçim veri kümesinin 3260 sol parça, 2740 sağ parça düğümüne ve bunlar arasında toplam 30000 ayrıta sahip daraltılmış bir veri kümesi kullanılmaktadır. Bu algoritmaların testinde 1747 sol parça, 1530 sağ parça düğümüne ve 4200 ayırta sahip, sentetik veri kümesinden elde edilmiş bir çizge daha kullanılmıştır.

Yaklaşık eşleme işlemi oldukça maliyetli bir işlem olduğu için yaklaşık eşleme işlemi için seçim veri kümesinin daraltılmış bir diğer altkümesi üzerinde sorgular gerçekleştirilmektedir. Bu daraltılmış çizge, sol parçasında 327 düğüm, sağ parçasında 452 düğüm ve 1368 ayrıttan oluşmaktadır. Gerçek veri kümesine ek olarak sentetik veri kümesinden elde edilen bir çizge üzerinde de yaklaşık eşleme algoritmaları test edilmiştir. Sentetik veri kümesinden elde edilen bu çizge 220 sol parça, 190 sağ parça ve 1000 ayrıttan oluşmaktadır.

5.1.3. Kullanılan Algoritmalar

Performans karşılaştırması için kullanılan algoritmalar seçilirken karşılaştırılacağı algoritma ile aynı problem varyasyonuna çözüm üretmesine ve aynı çözüm yaklaşımına sahip olmasına dikkat edilmektedir. Karşılaştırma için kullanılan tüm algoritmalara ait kodlar, algoritmayı oluşturan yazarlardan elde edilmektedir.

Geliştirilen dal-ve-sınır algoritması öncelikle VF2 algoritması ile performans karşılaştırma testine tabi tutulmaktadır. VF2 algoritması, birebir eşleme problemi için geliştirilen çözüm yaklaşımlarını kıyaslamada en yaygın kullanılan algoritmadır. İki uzaklıklı komşulukları, geliştirilen algoritmada olduğu gibi dikkate alıyor olması ve dal-ve-sınır tekniğini kullanıyor olması sebebiyle kıyaslama yapmak için uygun bir algoritmadır. Performans karşılaştırmasına dahil edilen diğer bir algoritma ise Turbo^{iso} algoritmasıdır. Bu algoritma, yakın zamanda geliştirilen dal-ve-sınır algoritmaları arasında, en düşük çalışma süreleri ile en yüksek performansı gösteren algoritmalardan biri olma özelliğini taşımaktadır. Turbo^{iso} algoritması da dallanma sırasında ön işlem ile elde ettiği verileri kullanarak uygun alt-yapılar üzerine dallanmaktadır ve bu yönü ile geliştirilen dal-ve-sınır algoritması ile benzerlik göstermektedir.

İndeks tabanlı algoritma literatürdeki yüksek performanslı GADDI algoritması ile performans açısından karşılaştırılmaktadır. GADDI algoritması tek bir büyük çizge içerisindeki sorgu çizgesi ile eşbiçimli alt-çizgeleri bulmayı hedeflemektedir.

Bu algoritma; komşu düğüm çiftlerini, algoritmanın NDS ismini verdiği bir uzaklık değeri ile indekslemektedir. Daha sonra sorgu çizgesi uygun ikililerin girişimi ile tekrar oluşturulmaktadır. Bu bölme, indeksleme ve girişim ile tekrar elde etme yaklaşımı geliştirilen indeks tabanlı algoritma ile oldukça benzer olduğu için bu algoritma karşılaştırma için uygun bir aday olmaktadır.

Yaklaşık eşleme problemi için geliştirilen algoritma TALE algoritması ile karşılaştırılmaktadır. Bu algoritma düğümleri yapısal ve anlamsal özelliklerine göre derecelendirerek indekslemektedir. Daha sonra en önemli düğümden başlayarak belirlenen benzerlik oranı sağlanana kadar mevcut çözüm genişletilmektedir. Her iki yöntem de kısmi bir çözümden başlayıp belirli bir benzerlik düzeyi sağlanana kadar bu kısmi çözümlerin genişletilmesi mantığına dayandığı için adil bir kıyaslama yapmak mümkündür.

5.2. Deneysel Sonuçlar

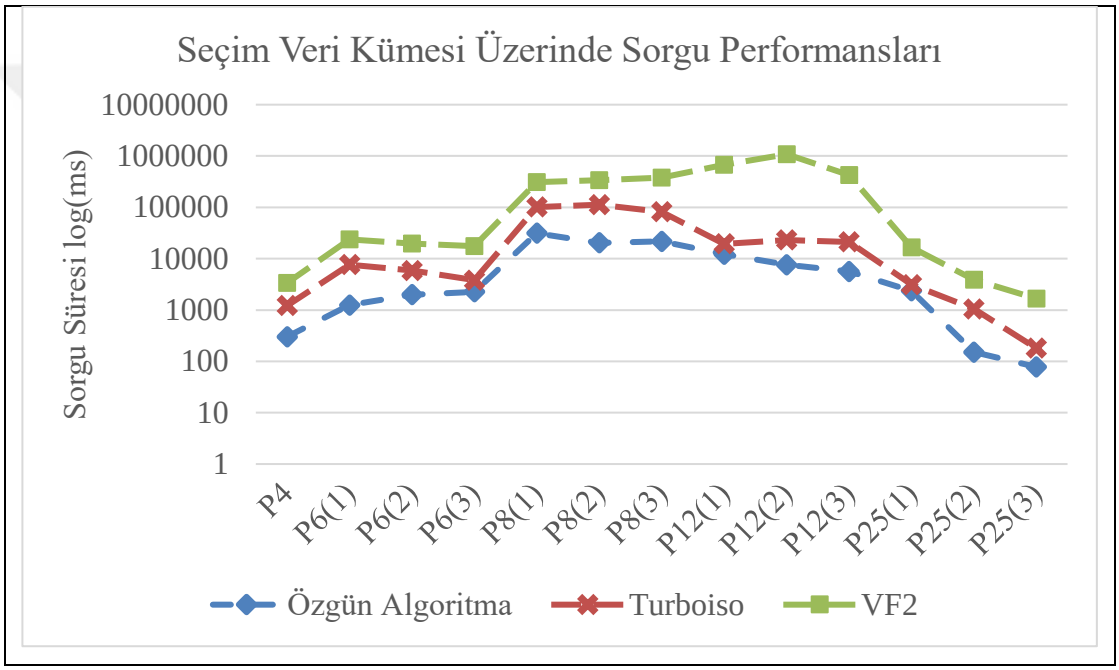
5.2.1. Dal-ve-sınır Algoritması

Geliştirilen dal-ve-sınır algoritması, VF2 ve Turbo^{iso} algoritmalarının farklı çizge boyutları üzerinde çalışma sürelerinin tespiti için bahsedildiği üzere üç farklı hedef çizge kullanılmıştır. Tüm bu algoritmalar birebir eşleme algoritması olduğu için aynı hedef çizge içerisinde aynı sayıda eşbiçimli alt-çizge bulmalıdır fakat geliştirilen özgün algoritma, bu tez çalışmasında tanımı yapıldığı üzere iki parçalı çizgeler üzerinde eşbiçimlilik sorgusu yaptığı için eksik sonuç üretmektedir. Çalışma sürelerini adil biçimde karşılaştırabilmek için özgün algoritma ile yapılan sorgularda, sorgu çizgesinin simetrikleri ile de sorgu gerçekleştirilmektedir. Her üç algoritma da Tablo 5.1'de gösterilen tüm sorgu çizgeleri ile her üç hedef çizge üzerinde alt-çizge eşbiçimlilik sorgusu gerçekleştirmektedir. Tablo 5.2'de bazı alt çizge eşbiçimlilik sorguları sonucu elde edilen sonuç sayıları gösterilmektedir.

Tablo 5.2: Bazı Sorgular Sonucu Elde Edilen Alt-çizge Sayıları.

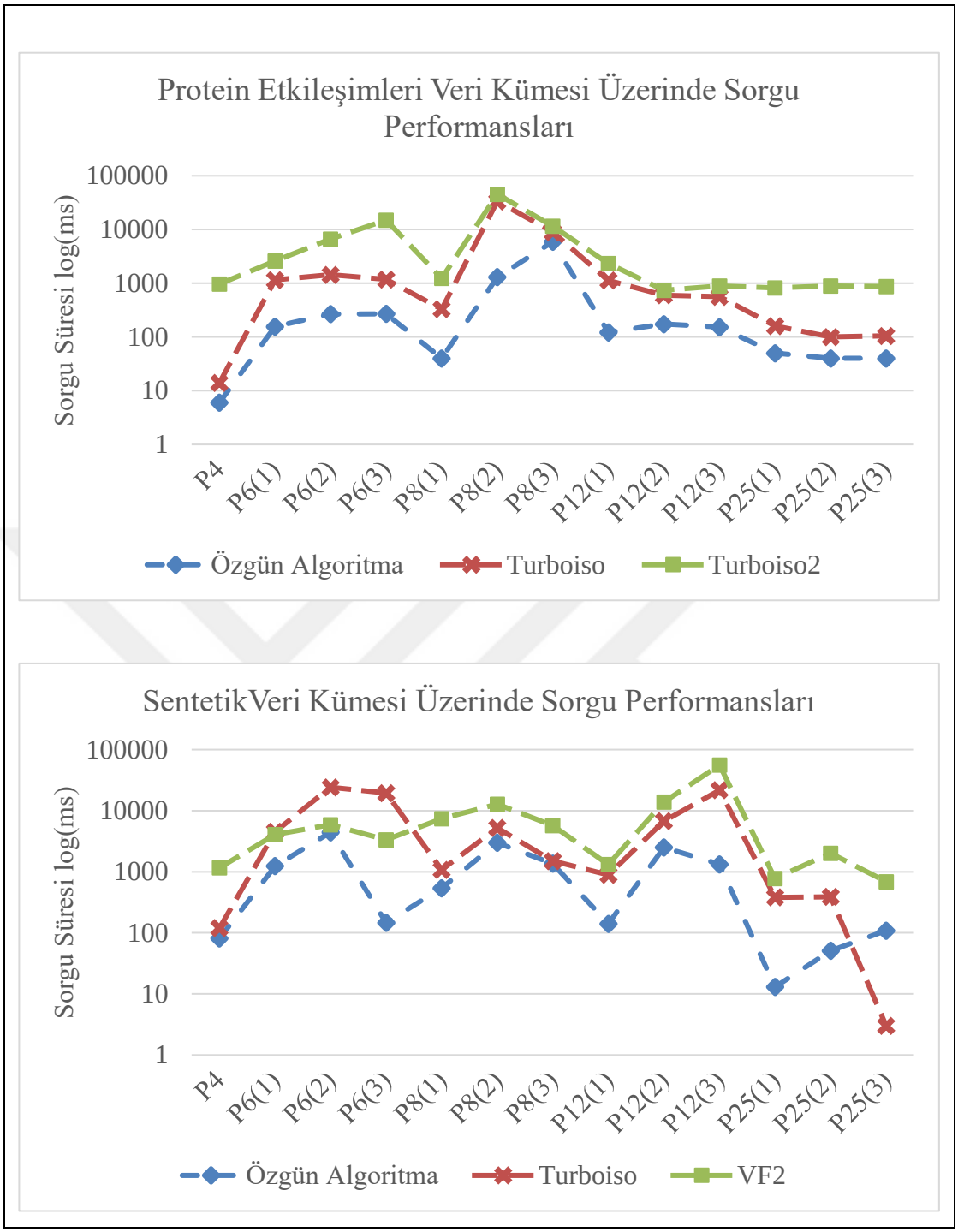
Veri Kümesi	P_4	$P_{6\text{-}3}$	$P_{8\text{-}3}$	$P_{12\text{-}3}$	$P_{25\text{-}3}$
Herpesviridae	927	558243	37984	2241	1
Seçim Verileri	89589	692750	114094	67	0
Sentetik Veri	324	191681	67432	1197	0

Tablo 5.1'de bildirilen sorgu sonuç sayıları, her üç çizgenin sadece boyut olarak değil yapısal olarak da çeşitlilik gösterdiğini ortaya koymaktadır. Sorgu sonuçları incelendiğinde 6 ve 8 düğüme sahip sorgu çizgelerinin en fazla sayıda bulunduğu görülmektedir. Hedef çizgeler, yoğun çizgeler olduğu için bu çizgelerin kombinasyonlarını bolca bulmak mümkündür. Diğer taraftan 25 düğüme sahip sorgu çizgesi oldukça özel bir sorgu çizgesi olduğu için üç çizgede toplam sadece 1 adet bulunmaktadır. Her üç algoritma da birebir eşleme algoritması olduğu için eşit sayıda ve birebir aynı sonuçları üretmektedir. Bu algoritmaların sorguları gerçekleştirmek için harcadıkları süreler Şekil 5.2 ve 5.3'de bildirilmektedir.



Şekil 5.2: Dal-ve-sınır algoritmalarının daraltılmış veri kümesi üzerindeki sorgu performansları.

Şekil 5.2 ve Şekil 5.3'de bildirilen çalışma süreleri açıkça göstermektedir ki geliştirilen özgün algoritma, farklı boyuta sahip hedef ve sorgu çizgeleri yapılan tüm sorgularda VF2 ve Turbo^{iso} algoritmasına çalışma süreleri bakımından üstünlük sağlamaktadır.



Şekil 5.3: Dal-ve-sınır algoritmalarının Herpesviridae protein etkileşim çizgesi (üstte) ve sentetik veri kümesi üzerindeki sorgu performansları.

Sorgu performansları tekrar incelendiğinde düğüm sayısı düşük olan sorgu çizgeleri ile yapılan sorgularda performans farkının çok daha fazla olduğu görülmektedir. Bu durumun sebebi geliştirilen algoritmanın tek bir parça üzerine dallanması ve bu dallanmaya ait uygun sonuçların hızlıca üretilebilmesinden kaynaklanmaktadır.

Her üç algoritma da güçlü budama teknikleri sayesinde sorgu çizgelerinin boyutu arttığında da sorgu yapma imkânı sağlamaktadır. Uygun olmayan sonuçlar her üç algoritma tarafından da hızlı bir şekilde budanabildiği için sorgu çizgesinin boyutu arttıkça çalışma süreleri kısalmakta ve aradaki performans farkı kapanmaktadır. Sorgu çizgeleri tekrar incelendiğinde, ayrıt yoğunluğu yüksek olan sorgu çizgeleri ile sonuca daha hızlı varıldığı söylenebilmektedir. Bu durumun sebebi, algoritmaların budama işlemi için düğüm derecelerinden faydalanmasıdır: Yoğun sorgu çizgelerindeki düğümlerin dereceleri daha büyük olacağı için aday sayıları daha düşüktür ve böylece sorgu daha hızlı tamamlanabilmektedir.

Sonuç olarak geliştirilen özgün dal-ve-sınır algoritması, literatürdeki yüksek performanslı VF2 ve Turbo^{iso} algoritmalarına kıyasla, iki parçalı çizgeler üzerinde 2 ile 500 kat arasında daha kısa sürede sorgu gerçekleştirebilmektedir.

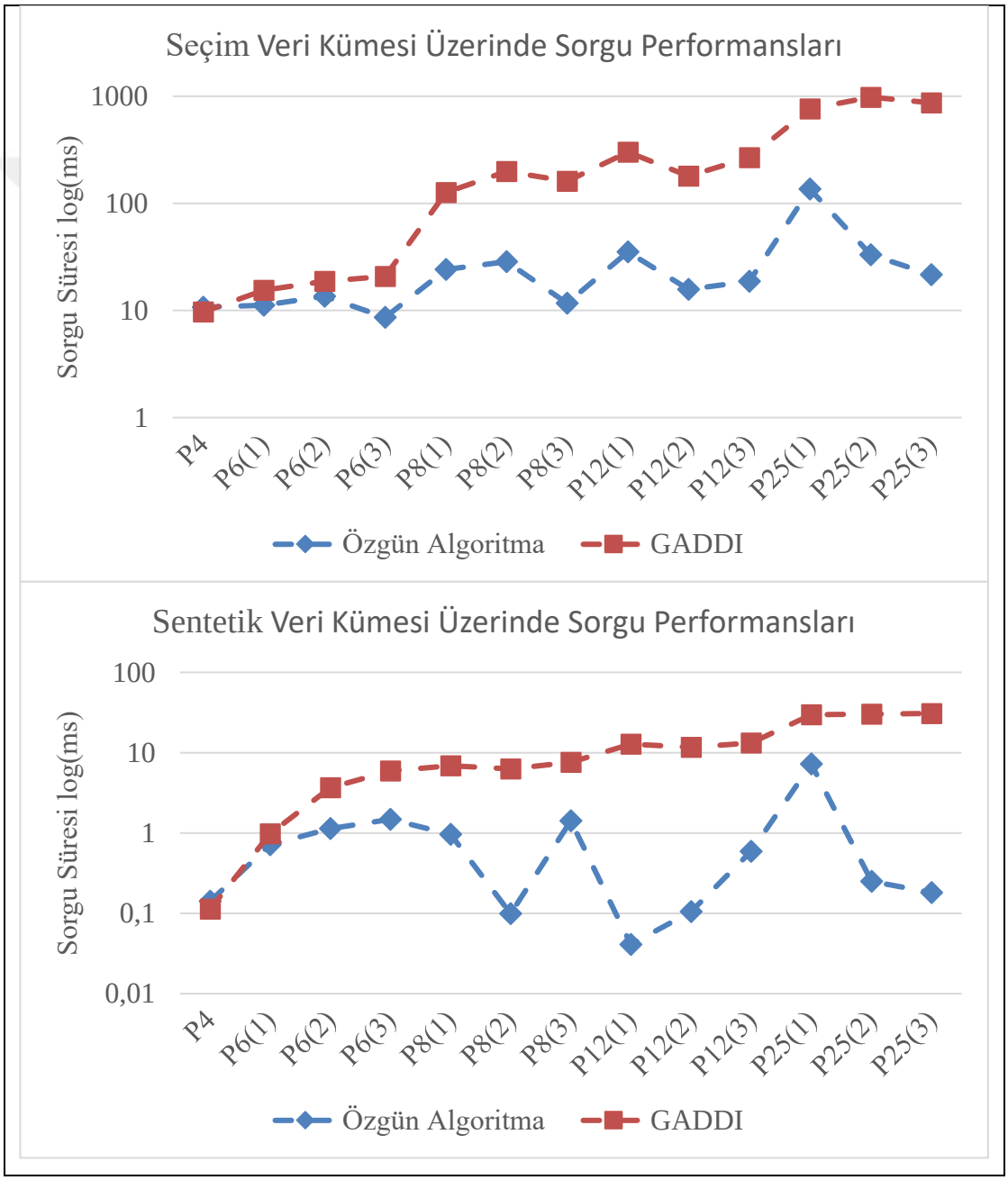
5.2.2. İndeks Tabanlı Algoritma

Özgün indeks tabanlı algoritma ve GADDI algoritması ile seçim verilerini içeren büyük çizge üzerinde ve sentetik olarak üretilen orta büyüklükteki çizge üzerinde alt-çizge eşbiçimlilik sorgusu gerçekleştirilmiştir. Tablo 5.1'de bildirilen sorgu çizgelerinin her biri ile sorgu gerçekleştirilerek performans sonuçları incelenmiştir. Geliştirilen indeks tabanlı algoritma için gerekli YOL oluşturulurken; 4, 6 ve 8 düğüme sahip sorgu çizgeleri için yeniden oluşturma listeleri kaba-kuvvet algoritması ile 12 ve 25 düğüme sahip sorgu çizgeleri için YOLlar açgözlü algoritma ile oluşturulmuştur. Algoritmaların, seçim veri kümesine ait indeks verilerini oluşturmak için harcadıkları süreler ve indeks veri sayıları Tablo 5.3'de bildirilmektedir:

Tablo 5.3: Algoritmaların seçim veri kümesine ait indeks verilerini oluşturma süreleri ve indeks sayıları.

Algoritma	Çalışma süreleri (sn)	İndeks Sayısı
Özgün Algoritma	72	$620x10^3$
GADDI	359	$5x10^3$

Tablo 5.3'de bildirilen çalışma süreleri, geliştirilen özgün algoritmanın indeks verilerini hızlı biçimde oluşturabildiğini göstermektedir. Bunun yanında oluşan indeks sayısı GADDI algoritmasına kıyasla oldukça fazladır. Bu durumun sebebi: GADDI algoritması her bir düğüm için sadece bir indeks verisi oluşturmaktadır. Geliştirilen indeks tabanlı algoritma iki uzunluktaki tüm yolları indekslediği için indeks sayısı oldukça fazla olmaktadır. Algoritmaların Tablo 5.1'de gösterilen sorgu çizgeleri ile yapılan sorgulara ait çalışma süreleri Şekil 5.4'de bildirilmektedir:



Şekil 5.4: İndeks tabanlı algoritmaların, sentetik ve seçim veri kümeleri üzerinde gerçekleştirdiği alt-çizge eşbiçimlilik sorgularına ait çalışma süreleri.

Her iki algoritmanın da artan sorgu çizge boyutları sebebiyle çalışma sürelerinin uzadığı görülmektedir. Bu durumun sebebi daha büyük sorgu çizgelerinin daha fazla girişim işlemine ihtiyaç duymasıdır. Şekil 5.4'e bakıldığında çalışma sürelerinin başlarda birbirine oldukça yakın olduğu görülmektedir. Geliştirilen özgün algoritma uygunsuz sonuçları daha ilk girişimden büyük ölçüde elediği için GADDI algoritmasına kıyasla, artan sorgu çizgesi boyutları ile daha iyi başa çıkmaktadır.

Tablo 5.4: İndeks Tabanlı algoritma ile seçim verilerine ait çizge üzerinde gerçekleştirilen bazı sorgulara dair istatistikler.

Sorgu Çizgesi	Girişim Sayısı	Sonuç Sayısı	Ortalama Aday Kazayağı Sayıları
P_4	1233373	695896	180×10^3
P_{6-3}	1914066	962750	40×10^3
P_{8-3}	2475621	114094	11×10^3
P_{12-3}	2619482	6287	5×10^3
P_{25-3}	3718260	2	750

Tablo 5.4'te geliştirilen algoritmanın daha iyi anlaşılması için gerçekleştirilen sorgulara dair bazı detay bilgilere yer verilmektedir. Tablo dikkatli incelendiğinde, beklendiği üzere, yüksek dereceli düğümlere sahip çizgelerdeki ortalama aday kazayağı sayısı azalmaktadır. Diğer yandan sorgu çizgelerinin boyutu ile YOL'ların uzunluğu arasında doğru orantı bulunduğu için gerekli girişim sayısı beklendiği kadar azalmamaktadır.

Sonuç olarak geliştirilen indeks tabanlı algoritmanın gerçekleştirdiği sorgulara ait veriler değerlendirildiğinde: Geliştirilen algoritma, GADDI algoritmasına çalışma süresi bakımından 40 kata kadar daha üstündür. Diğer taraftan geliştirilen indeks tabanlı algoritmada, tüm iki uzunluklu yolları indekslendiği için oluşan indeks sayısı oldukça fazladır. İndeks verilerinin sayısı arttıkça ihtiyaç duyulan hafıza alanı da oldukça artmaktadır. Algoritmanın indeks mekanizması, veri yapılarının hafızada tutulması üzerine olduğu için, artan veri yapısı boyutları, büyük çizgelere ait indeks verilerini oluşturmayı ve hafızada tutmayı zorlaştırmaktadır.

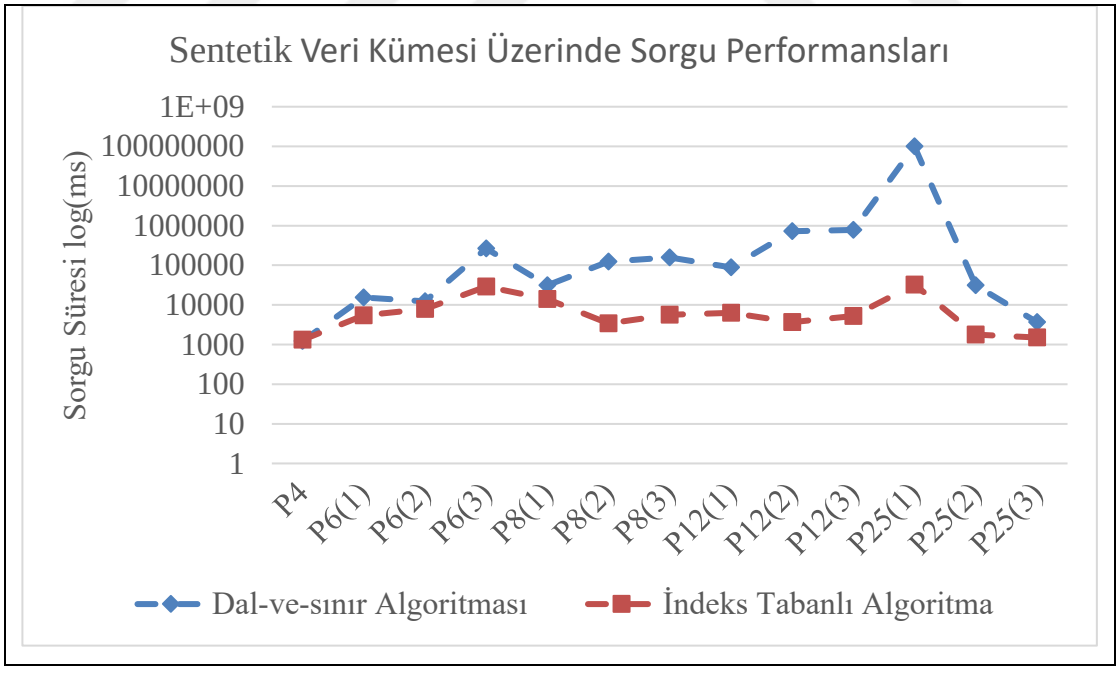
5.2.3. Özgün Algoritmaların Karşılaştırılması

Tez kapsamında geliştirilen özgün algoritmalar birbirleri ile de performans testlerine tabi tutulmuştur. Bu işlem için Tablo 5.5'te gösterilen hedef çizgeler üzerinde Tablo 5.1'de gösterilen tüm sorgu çizgeleri çalıştırılarak performans değerleri elde edilmiştir.

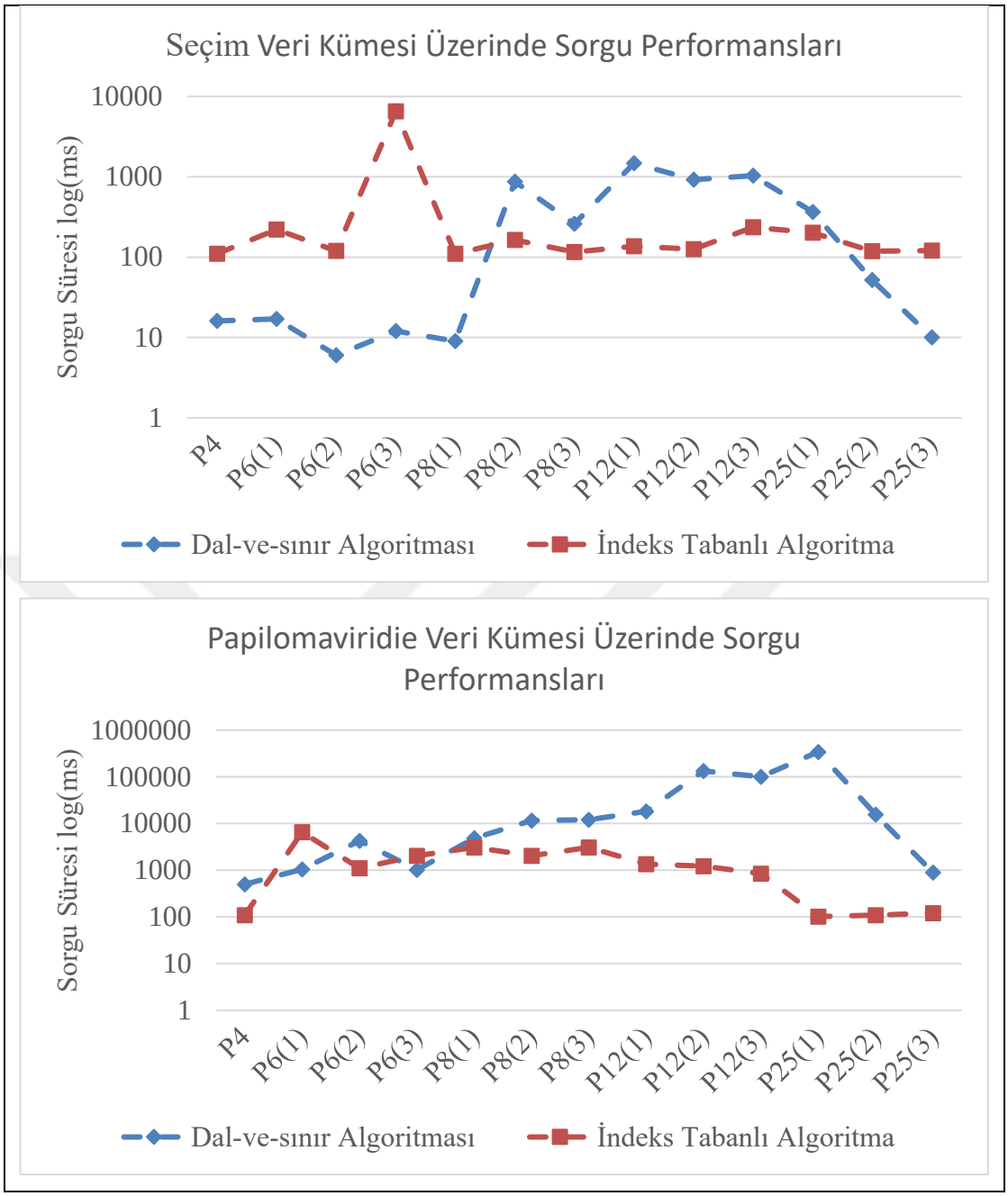
Tablo 5.5: Özgün Algoritmaları Karşılaştırmada Kullanılan Veri Kümeleri.

Veri Kümesi	Sol Parça Düğüm Sayısı	Sağ Parça Düğüm Sayısı	Ayrıt Sayısı
Papilomaviridie	120	760	1100
Seçim Verileri	196	211	650
Sentetik Veri	720	1600	4367

Karşılaştırma testleri büyüklükleri ve yoğunlukları farklı bu üç veri kümesinden elde edilen çizgeler üzerinde yapılmıştır. Elde edilen çalışma süreleri Şekil 5.5 ve 5.6'da bildirilmektedir.



Şekil 5.5: Özgün algoritmaların büyük hedef çizge üzerindeki sorgu süreleri.

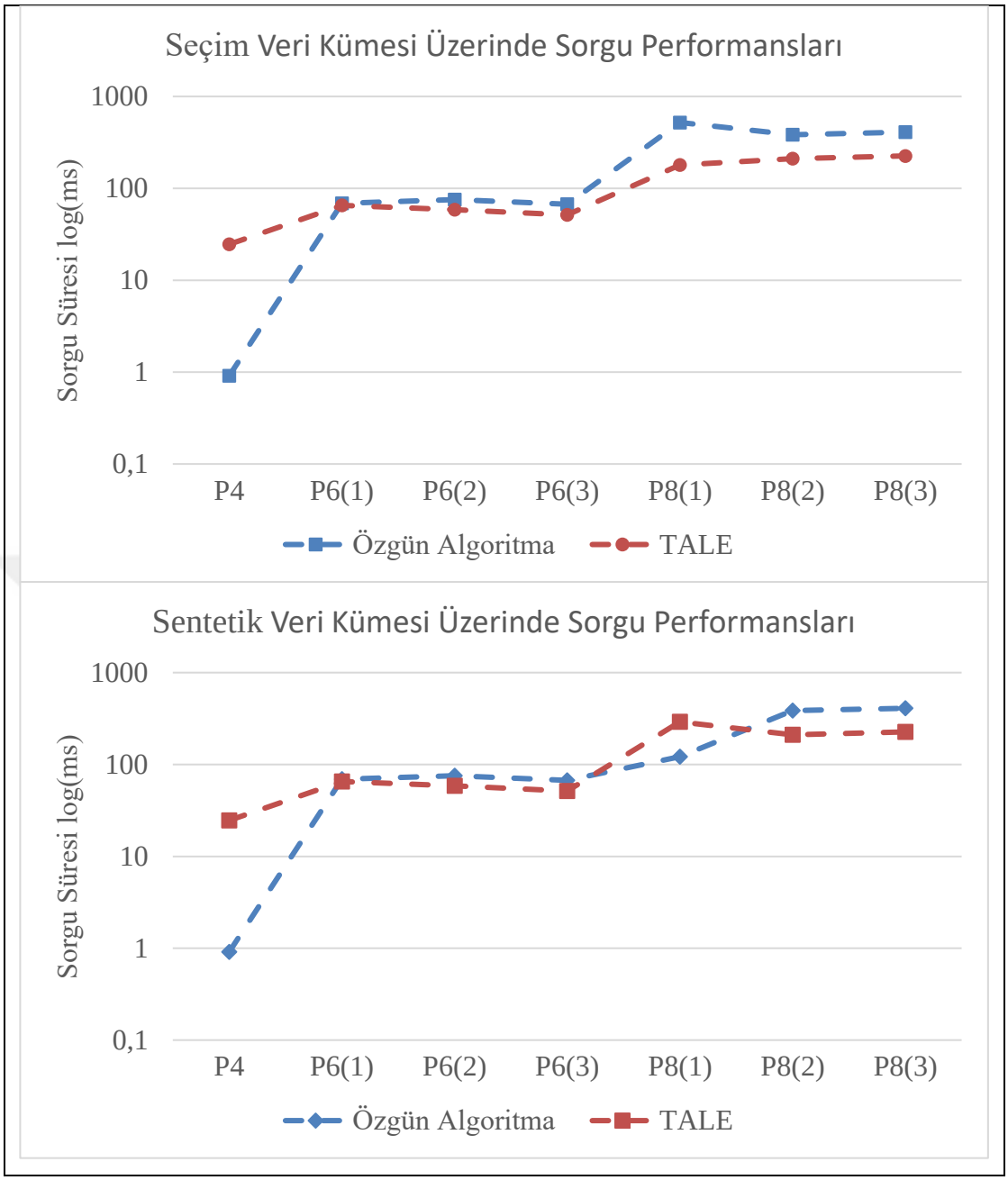


Şekil 5.6: Özgün algoritmaların orta büyüklükte çizge üzerindeki sorgu süreleri.

Performans sonuçları göstermektedir ki geliştirilen dal-ve-sınır algoritması bir parçasında görece diğer parçaya göre az düğüm bulunan orta ölçekli çizgelerde indeks tabanlı algoritmaya göre daha iyi performans göstermektedir. Sonuçlar analiz edildiğinde ortaya çıkan bir diğer durum ise düğüm sayısı az olan sorgu çizgelerinde performans farklarının daha az olduğu, sorgu çizgesinin düğüm sayısı arttıkça performans farkının da arttığı gözlemlenmektedir.

5.2.4. Yaklaşık Eşleme Algoritması

Tez kapsamında geliştirilen yaklaşık eşleme algoritması, alandaki yüksek performanslı yaklaşık eşleme algoritmalarından TALE algoritması ile çalışma süreleri ve üretilen sonuç sayıları üzerinden karşılaştırılmaktadır. TALE algoritması, büyük tek bir çizge üzerinde sorgu yapma hedefi ile geliştirilmiştir. Algoritmalar yaklaşık eşleme sorgusunu seçim veri kümesinin daraltılmış bir alt kümesi ve sentetik veri kümesinden elde edilen çizgeler üzerinde gerçekleştirmektedir. Sorgu için Tablo 5.1'de gösterilen 4, 6 ve 8 düğümlü sorgu çizgeleri kullanılmaktadır. Bu sorgulara yaklaşık sorgu özelliği kazandırmak için bir ayrıtın eksikliğine müsaade edilmektedir. TALE algoritması, benzerlik oranı üzerinden yaklaşık eşleme yaptığı için 4, 6 ve 8 düğümlü sorgu çizgeleri için sırasıyla %80, %83.33 ve %87.50 benzerlik oranları, algoritmaya parametre olarak verilmektedir. Şekil 5.7'de TALE algoritması ile özgün yaklaşık eşleme algoritmasının çalışma süreleri bildirilmektedir.



Şekil 5.7: Yaklaşık eşleme algoritmalarına ait sorgu süreleri.

Elde edilen çalışma süreleri incelendiğinde çalışma süreleri bakımından birbirine yakın performans sonuçlarına sahip olduğu gözlemlenmektedir. Diğer taraftan, bu algoritmalar yaklaşık eşleme algoritması olduğu için ürettiği sonuç sayılarına da bakılmalıdır. Tablo 5.6'da algoritmaların seçim verisi üzerinde yaptığı sorgular sonucu ürettiği sonuç sayıları gösterilmektedir.

Tablo 5.6: Yaklaşık eşleme algoritmalarının ürettiği sonuç sayıları.

Algoritma	P_4	P_{6-1}	P_{6-2}	P_{6-3}	P_{8-1}	P_{8-2}	P_{8-3}
Özgün algoritma	$2x10^6$	$6,4x10^6$	$6,6x10^6$	$7,2x10^6$	$32x10^6$	$24,8x10^6$	$20x10^6$
TALE	$2x10^6$	$3,7x10^6$	$4,5x10^6$	$4,6x10^6$	$10,3x10^6$	$8,5x10^6$	$11,2x10^6$

Tablo 5.6'da bildirilen veriler açıkça göstermektedir ki geliştirilen yaklaşık eşleme algoritması sorgu çizge boyutu ile doğru orantılı olarak daha fazla sonuç üretmektedir. Bu durumun sebebi geliştirilen özgün algoritmanın, müsaade edilen sınırlar içerisindeki tüm sonuçları bulmayı garanti etmesidir. TALE algoritması, bulduğu sonuçların müsaade edilen benzerlik sınır değerinin üzerinde olduğunu garanti etse de bu sınırın üzerindeki tüm sonuçların üretileceğini garanti etmemektedir. Elde edilen her iki karşılaştırma sonuçları beraber değerlendirildiğinde: Geliştirilen özgün algoritma, sonuçları üretmek için TALE algoritmasından daha fazla işlem gücüne ihtiyaç duymaktadır. Geliştirilen özgün algoritma olası tüm yaklaşık eşlemeleri bulmayı garanti etmektedir ve bu sebeple yaklaşık %50 daha fazla sonuç üretmektedir.

6. SONUÇ ve GELECEK ÇALIŞMALAR

Tez çalışmasına başlarken iki parçalı çizgelerin kendine özgü yapısından faydalanıldığında, alt-çizge eşbiçimlilik problemine yüksek performanslı çözüm yöntemleri geliştirilebileceği öngörülmüştür. Bu amaçla iki parçalı alt-çizge eşbiçimlilik probleminin birebir ve yaklaşık eşleme varyasyonlarına özgün çözüm algoritmaları geliştirilmiştir. Geliştirilen bu özgün algoritmaların hem teorik hem de deneysel olarak yüksek performans gösterdikleri yapılan analiz ve deneyler ile saptanmıştır.

Birebir eşleme problemine geliştirilen dal-ve-sınır algoritması, iki parçalı yapıyı, dallanma adaylarının seçiminde ve uygunsuz dallanmaların oluşmasının önüne geçmede kullanmaktadır. Sorgu çizgesi düğümlerine aday seçimi yapılırken sadece sorgu düğümü ile aynı parçadaki hedef düğümler arasından adaylar seçilmektedir. Bu durum olası dallanma sayısını büyük ölçüde azaltmaktadır. Diğer taraftan iki parçalı çizgelerde iki uzunluktaki yollar ile birbirine bağlı olan düğümlerin aynı parçada olması gerekliliği, uygun sonuç üretmesi imkânsız dallanmaları daha oluşmadan budamayı sağlamaktadır. Bu budama işleminin dallanma uzayını oldukça düşürdüğü gözlemlenmektedir. Dallanma uzayındaki bu daralma algoritmanın performansına büyük katkı sağlamaktadır.

İndeks tabanlı birebir eşleme algoritması, iki parçalı yapıyı oldukça iyi karşılayan kazayağı indeks yapısını kullanmaktadır. Kazayağı indeks yapısı sayesinde etkili bir filtreleme gerçekleştirmek mümkün olmaktadır. Sorgu çizgesini oluşturan kazayağlarına uygun adayları elde etmek için hedef çizge filtrelenirken sadece tepe düğümü aynı parçada olan kazayağları aday olarak belirlenmektedir. Bu sayede sorgu çizgesi yeniden oluşturulurken kullanılacak aday sayısı oldukça azaldığı gözlemlenmektedir ve böylece algoritmanın performansı artmaktadır.

Yaklaşık eşleme problemine geliştirilen algoritma, indeks tabanlı algoritma gibi kazayağı indeks yapısını kullandığı için bu algoritmanın yararlandığı tüm özelliklerden yaklaşık eşleme algoritması da faydalanmaktadır.

Tez çalışması kapsamında geliştirilen özgün algoritmaların bazı alt problemlere çözüm üretilirken her zaman optimal sonuçları veremeyen yöntemler kullanılmıştır. Bu problemlerden biri en az maliyete sahip YOLun bulunmasıdır. Kullanılan açgözlü yaklaşım daha önce değinildiği üzere her zaman optimal YOLu bulamamaktadır. Diğer yandan en iyi YOL kıstası da geliştirmeye açıktır: En az sayıda aday girişimine

sahip YOLdan daha hızlı sonuç üretebilen bir YOL bulunabilir. Bu problem gelecekte çözülmesi gereken açık bir problemdir.

Dal-ve-sınır algoritmasının üzerinde dallanma gerçekleştirdiği parçanın seçilmesi de yine üzerinde iyileştirilme yapılabilecek problemlerdendir. Mevcut algoritma daima sol parça üzerinde dallanma, sağ parça üzerinde filtreleme yaparak sonuçları elde etmektedir. Diğer taraftan, veri kümesinin öz niteliklerine göre en az sayıda dallanma ile sonuca ulaşılabilmesi için hangi parça üzerinde dallanma, hangi parça üzerinde filtreleme yapılacağı problemi de yine açık bir problemdir.

Bir diğer geliştirmeye açık problem yaklaşık eşleme yapmak için ihtiyaç duyulan yeniden oluşturma listelerinin seçimidir. Özgün yaklaşık eşleme algoritması geliştirilirken iki farklı strateji ile bu listeler oluşturulmaktadır ve bu yaklaşımlardan birinin diğerine oldukça üstün olduğu görülmektedir. Kullanılan yüksek performanslı seçim stratejisi, en az maliyetle yaklaşık eşlemeyi garanti etmemektedir ve bu sebeple yaklaşık eşleme için gerekli YOLların bulunması da yine açık bir problemdir.

KAYNAKLAR

Afrati F. N., Fotakis D., Ullman J. D., (2013), "Enumerating Subgraph Instances Using Map-reduce", In 2013 IEEE 29th International Conference on Data Engineering , 62-73, Brisbane, QLD, Australia, 8-12 April.

Hagberg A. A., Schult D. A., Swart P. J., (2008), "Exploring Network Structure, Dynamics and Function Using Networkx", In Proceedings of the 7th Python in Science Conference (SciPy2008), 11-15, Pasadena, CA, USA, 19–24 August.

Asiler M., Yazıcı A., (2017), "BB-Graph: A Subgraph Isomorphism Algorithm for Efficiently Querying Big Graph Databases", Master Thesis, Middle East Technical University.

Asratian A., Denley T., Häggkvist R., (1998), "Bipartite Graphs and Their Applications", 1st Edition, Cambridge University Press.

Bi F., Chang L., Lin X., Qin L., Zhang W., (2016), "Efficient Subgraph Matching by Postponing Cartesian Products", In Proceedings of the 2016 International Conference on Management of Data, 1199-1214, New York, USA, 26-30 June.

Bishop C. M., (2006), "Pattern Recognition and Machine Learning", 1st Edition, Springer.

Bonnici V., Giugno R., Pulvirenti A., Shasha D., Ferro A., (2013), "A Subgraph Isomorphism Algorithm and Its Application to Biochemical Data", BMC Bioinformatics, 14(7), 13.

Brandes U., Eiglsperger M., Herman I., Himsolt M., Marshall M. S., (2001), "GraphML Progress Report Structural Layer Proposal", In International Symposium on Graph Drawing, 501-512, Vienna, Austria, 23-26 September.

Cai D., Shao Z., Han J., (2005), "Community Mining From Multi-Relational Networks", European Conference on Principles of Data Mining and Knowledge Discovery, 445-452, Porto, Portugal, 3-7 October.

Carletti V., Foggia P., Vento M., (2015), "VF2 Plus: An Improved Version of VF2 For Biological Graphs", In International Workshop on Graph-Based Representations in Pattern Recognition, 168-177, Vienna, Austria, 13-15 May.

Carletti V., Foggia P., Saggese A., Vento M., (2017), "Introducing VF3: A New Algorithm For Subgraph Isomorphism", In International Workshop on Graph-Based Representations in Pattern Recognition , 128-139, Anacapri, Italy, 16-18 May.

Chen C., Yan X., Yu P. S., Han, J., (2007), "Towards Graph Containment Search and Indexing", In Proceedings of The 33rd International Conference on Very Large Databases, 926-937, Viyana, Austria, 23-27 September.

Cheng J., Ke Y., Ng W., Lu A., (2007), "Fg-index: Towards Verification-Free Query Processing on Graph Databases", In Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, 857-872, Beijing, Chaina, 12-14 June.
Cheng J., Yu J. X., Ding B., Yu P. S., Wang H., (2008), "Fast Graph Pattern Matching", International Conference on Data Engineering, 913-922, Cancun, Mexico, 7-12 April.

Čibej U., Mihelič J., (2015), "Improvements to Ullmann's Algorithm For The Subgraph Isomorphism Problem", International Journal of Pattern Recognition and Artificial Intelligence, 29(07), 1550025.

Cook S., (1971), "The Complexity of Theorem-Proving Procedures", In Proceedings of The Third Annual ACM Symposium on Theory of Computing, 151-158, NY, USA, 13-16 May.

Cordella L. P., Foggia P., Sansone C., Vento M., (2004), "A (sub) Graph Isomorphism Algorithm For Matching Large Graphs", IEEE Transactions on Pattern Analysis and Machine Intelligence, 26(10), 1367-1372.

Corneil D., Gotlieb C., (1970), "An Efficient Algorithm For Graph Isomorphism", Journal of the ACM (JACM), 17(1), 51-64.

Di Natale R., Ferro A., Shasha D., (2010), "Sing: Subgraph Search in Non-Homogeneous Graphs", BMC Bioinformatics, 11(1), 96.

Durmuş T. S., Çakır T., Ardıç E., Sayılırbaş A. S., Konuk G., Konuk M., Sevilgen F. E., (2013), "PHISTO: Pathogen–Host Interaction Search Tool", Bioinformatics, 29(10), 1357-1358.

Fankhauser S., Riesen K., Bunke H., Dickinson P., (2012), "Suboptimal Graph Isomorphism Using Bipartite Matching", International Journal of Pattern Recognition and Artificial Intelligence, 26(6), 1250013.

Foggia P., Percannella G., Vento M., (2014), "Graph Matching and Learning In Pattern Recognition In The Last 10 Years", International Journal of Pattern Recognition and Artificial Intelligence, 28(01), 1450001.

Gao X., Xiao B., Tao D., Li X., (2010), "A Survey of Graph Edit Distance", Pattern Analysis and Applications, 13(1), 113-129.

Giugno R., Shasha D., (2002), "Graphgrep: A Fast and Universal Method For Querying Graphs. In Object Recognition Supported by User Interaction For Service Robots", 112-115, Quebec, Canada, 11-15 August.

Giugno R., Bonnici V., Shasha D., (2013), "Grapes: A Software For Parallel Searching on Biological Graphs Targeting Multi-Core Architectures", PloS One, 8(10), e76911.

Gülsoy G., Kahveci T., (2011), "RINQ: Reference-Based Indexing For Network Queries", Bioinformatics, 27(3), i149-i158.

Han W. S., Lee J. H., (2013), "Turbo^{iso}: Towards Ultrafast and Robust Subgraph Isomorphism Search In Large Graph Databases", In Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, 337-348, New York, USA, 13-17 June.

He H., Singh A. K., (2008), "Query Language and Access Methods For Graph Databases", In Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, 405-418, Vancouver, Canada, 10-12 June.

He H., Singh K., (2006), "Closure-Tree: An Index Structure For Graph Queries", In 22nd International Conference on Data Engineering (ICDE'06), 38, Atlanta, GA, USA, 3-7 April.

Hong L., Zou L., Lian X., Philip S. Y., (2015), "Subgraph Matching with Set Similarity In A Large Graph Database", IEEE Transactions on Knowledge and Data Engineering", 27(9), 2507-2521.

Hopcroft J., Karp R., (1973), "An $n^{5/2}$ Algorithm For Maximum Matchings In Bipartite Graphs", SIAM Journal on Computing, 2(4), 225-231.

Jiang H., Wang H., Philip S. Y., Zhou S., (2007), "Gstring: A Novel Approach For Efficient Search In Graph Databases", In 2007 IEEE 23rd International Conference on Data Engineering, 566-575, Istanbul, Turkey, 11-15 April.

Jin L., Chen Y., Wang T., Hui P., Vasilakos A. V., (2013), "Understanding User Behavior In Online Social Networks: A Survey", IEEE Communications Magazine, 51(9), 144-150.

Jin W., Yang J., (2011), "A Flexible Graph Pattern Matching Framework via Indexing", In International Conference on Scientific and Statistical Database Management, 293-311, Portland, OR, USA, 20-22 Jully.

Khan A., Li N., Yan X., Guan Z., Chakraborty S., Tao S., (2011), "Neighborhood Based Fast Graph Search In Large Networks", In Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data, 901-912, Athens, Greece, 12-16 June.

Khan A., Wu Y., Aggarwal C. C., Yan X., (2013), "Nema: Fast Graph Search with Label Similarity", In Proceedings of the VLDB Endowment, 181-192, Riva del Garda, Italy, 26-31 August.

Kijima S., Otachi Y., Saitoh T., Uno T. (2012), "Subgraph Isomorphism In Graph Classes", Discrete Mathematics, 312(21), 3164-3173.

Lacroix V., Fernandes C. G., Sagot M. F., (2006), "Motif Search In Graphs: Application to Metabolic Networks", IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB), 3(4), 360-368.

Larrosa J., Valiente G., (2002), "Constraint Satisfaction Algorithms For Graph Pattern Matching", Mathematical Structures In Computer Science, 12(4), 403-422.

Lee J., Han W. S., Kasperovics R., Lee J. H., (2012), "An In-Depth Comparison of Subgraph Isomorphism Algorithms In Graph Databases", In Proceedings of the VLDB Endowment, 133-144, Istanbul, Turkey, 27-31 August.

Leskovec J., Huttenlocher D., Kleinberg J., (2010), "Signed Networks In Social Media", In Proceedings of the SIGCHI Conference on Human Factors In Computing Systems, 1361-1370, Atlanta, GA, 1-4 April.

Liang Y., Zhao P., (2017), "Similarity Search In Graph Databases: A Multi-Layered Indexing Approach", In 2017 IEEE 33rd International Conference on Data Engineering (ICDE), 783-794, San Diego, California, USA, 19-22 April.

Lin W., Xiao X., Cheng J., Bhowmick S. S., (2011), "Efficient Algorithms For Generalized Subgraph Query Processing", In Proceedings of the 21st ACM International Conference on Information and Knowledge Management, 325-334, Maui, Hawaii, USA, 29-30 October.

Liu J., Lee Y., (2001), "Graph-Based Method For Face Identification From A Single 2D Line Drawing", IEEE Transactions on Pattern Analysis and Machine Intelligence, 23(10), 1106-1119.

Messmer T., Bunke H., (1995), "Subgraph Isomorphism Detection In Polynomial Time", 1st Edition, Springer.

Messmer T., Bunke H., (1998), "A New Algorithm For Error-Tolerant Subgraph Isomorphism Detection", IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(5), 493-504.

Micale G., Pulvirenti A., Ferro A., Giugno R., Shasha D., (2019), "Fast Methods For Finding Significant Motifs on Labelled Multi-Relational Networks", Journal of Complex Networks, 7 (6), 817-837.

Mongiovi M., Giugno R., Pulvirenti A., Ferro A., Sharan R., (2010), "Sigma: A Set-Cover-Based Inexact Graph Matching Algorithm", Journal of Bioinformatics and Computational Biology, 8(2), 199-218.

Nabti C., Seba H., (2016), "Subgraph Isomorphism Search In Massive Graph Databases", In The International Conference on Internet of Things and Big Data, Roma, Italy, 23-25 April.

Nilsson N. J., (1980), "Principles of Artificial Intelligence", 1st Edition, Springer-Verlag.

Peng Y., Fan Z., Choi B., Xu J., Bhowmick S. S., (2014), "Authenticated Subgraph Similarity Search In Outsourced Graph Databases", IEEE Transactions on Knowledge and Data Engineering, 27(7), 1838-1860.

Pienta R., Tamersoy A., Tong H., Chau D. H., (2014), "MAGE: Matching Approximate Patterns In Richly-Attributed Graphs", In 2014 IEEE International Conference on Big Data (Big Data), 585-590, Washington DC, USA, 27-30 October.

Ren X., Wang J., (2015), "Exploiting Vertex Relationships In Speeding Up Subgraph Isomorphism Over Large Graphs", Proceedings of the VLDB Endowment, 8(5), 617-628.

Shang H., Lin X., Zhang Y., Yu J. X., Wang W., (2010), "Connected Substructure Similarity Search", In Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data , 903-914, Indianapolis, IN, USA, 6-11 June.

Shang H., Zhang Y., Lin X., Yu J. X., (2008), "Taming Verification Hardness: An Efficient Algorithm For Testing Subgraph Isomorphism", Proceedings of the VLDB Endowment, 1(1), 364-375.

Solnon C., (2010), "Different-Based Filtering For Subgraph Isomorphism", Artificial Intelligence, 174(12-13), 850-864.

Stauffer M., Tschachtli T., Fischer A., Riesen K., (2017), "A Survey on Applications of Bipartite Graph Edit Distance", In International Workshop on Graph-Based Representations in Pattern Recognition , 242-252, Anacapri, Italy, 16-18 May.

Tian Y., Patel J., (2008), "TALE: A Tool For Approximate Large Graph Matching", In 2008 IEEE 24th International Conference on Data Engineering , 963-972, Cancun, Mexico, 7-12 April.

Tian Y., McEachin R., Santos C., States D. J., Patel J. M., (2006), "SAGA: A Subgraph Matching Tool For Biological Graphs", Bioinformatics, 23(2), 232-239.

Ullmann J. R., (1976), "An Algorithm For Subgraph Isomorphism", Journal of the ACM (JACM), 23(1), 31-42.

Ullmann R., (2010), "Bit-Vector Algorithms For Binary Constraint Satisfaction and Subgraph Isomorphism", Journal of Experimental Algorithmics (JEA), 15, 1-6.

Vento M., (2015), "A Long Trip In The Charming World of Graphs For Pattern Recognition", Pattern Recognition, 48(2), 291-301.

Willett P., Barnard J. M., Downs G. M., (1998), "Chemical Similarity Searching", Journal of Chemical Information and Computer Sciences, 38(6), 983-996.

Xie Y., Yu P. S., (2011), "CP-Index: On The Efficient Indexing of Large Graphs", In Proceedings of the 20th ACM International Conference on Information and Knowledge Management, 1795-1804, Glasgow, Scotland, UK, 24-28 October.

Yan X., Yu P., (2005), "Substructure Similarity Search In Graph Databases", In Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, 766-777, Baltimore, Maryland, USA, 14-16 June.

Yan X., Yu P., Han J., (2004), "Graph Indexing: A Frequent Structure-Based Approach", In Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, 335-346, Paris, France, 13-18 June.

Yuan Y., Wang G., Chen L., Wang H., (2012), "Efficient Subgraph Similarity Search on Large Probabilistic Graph Databases", Proceedings of the VLDB Endowment, 5(9), 800-811.

Yuan Y., Wang G., Wang H., Chen L., (2011), "Efficient Subgraph Search Over Large Uncertain Graphs", Proc. VLDB Endowment, 4(11), 876-886.

Zampelli S., Deville Y., Dupont P., (2005), "Approximate Constrained Subgraph Matching", In International Conference on Principles and Practice of Constraint Programming, 832-836, Sitges, Spain, 1-5 October.

Zampelli S., Deville Y., Solnon C., (2010), "Solving Subgraph Isomorphism Problems With Constraint Programming", Constraints, 15(3), 327-353.

Zhang J., Kwong S., Jia Y., Wong K. C., (2017), "NSSRF: Global Network Similarity Search With Subgraph Signatures and Its Applications", Bioinformatics, 33(11), 1696-1702.

Zhang S., Hu M., Yang J., (2007), "Treepi: A Novel Graph Indexing Method", In 2007 IEEE 23rd International Conference on Data Engineering, 966-975, Istanbul, Turkey, 11-15 April.

Zhang S., Li S., Yang J., (2009), "GADDI: Distance Index Based Subgraph Matching In Biological Networks", In Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, 192-203, Saint Petersburg, Rusia, 27-30 March.

Zhang S., Yang J., Jin W., (2010), "SAPPER: Subgraph Indexing and Approximate Matching In Large Graphs", Proceedings of the VLDB Endowment, 3(1-2), 1185-1194.

Zhao P., Han J., (2010), "On Graph Query Optimization In Large Networks", Proceedings of the VLDB Endowment, 3(1-2), 340-351.

Zhao P., Yu J. X., Yu P. S., (2007), "Graph Indexing: Tree+ Delta<= Graph", In Proceedings of the 33rd International Conference on Very Large Databases, 938-949, Vienna, Austria, 23-27 September.

Zou L., Chen L., Özsu M. T., (2009), "Distance-Join: Pattern Match Query In A Large Graph Database", Proceedings of the VLDB Endowment, 2(1), 886-897.

Zou L., Chen L., Yu J. X., Lu Y., (2008), "A Novel Spectral Coding In A Large Graph Database", In Proceedings of the 11th International Conference on Extending Database Technology: Advances In Database Technology, 181-192, Nantes, France, 25-29 April.

ÖZGEÇMİŞ

Mehmet Burak Koca 1991 yılında Erzincan'da doğdu. İlk ve orta dereceli okulları Erzincan'da tamamladıktan sonra Yalova Üniversitesi Bilgisayar Mühendisliği Bölümünde lisans eğitimine başladı. 2015 Yılında Bilgisayar Mühendisliği Bölümünden lisans derecesini alarak Bilgisayar Mühendisi unvanını kazandı. 2016 ve 2017 yıllarında çeşitli yazılım firmalarında stajyer ve yazılımcı olarak çalıştı. 2017 yılı ocak ayında T.C. Uludağ Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalında başladığı yüksek lisans eğitimini 2018 Şubat ayında yarıda bırakarak T.C. Gebze Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalında yüksek lisans eğitimine başladı. 2018 Şubat ayından beri GTÜ Bilgisayar Mühendisliği Bölümünde araştırma görevlisi olarak çalışmaktadır.

EKLER

Ek A: Tez Çalışması Kapsamında Yapılan Yayınlar

Koca M. B., Sevilgen F. E., (2019), "A Novel Approach for Subgraph Isomorphism Problem on Bipartite Graphs", In 2019 27th Signal Processing and Communications Applications Conference (SIU), 1-4, Sivas, Turkey, 24-26 April.

Ek B: Terimler Sözlüğü

Tablo B1.1: Tez çalışmasında Türkçe olarak kullanılan İngilizce terimler.

Türkçe Terim	İngilizce Terim
Alt-çizge	Subgraph
Birebir Eşleme	Exact Matching
Çizge	Graph
Çizge Benzetme Uzaklığı	Graph Edit Distance
Dal-ve-sınır Algoritması	Branch-and-bound Algorithm
Eşbiçimlilik	Isomorphism
Girişim İşlemi	Join
İki Parçalı Çizge	Bipartite Graph
İndüklenmiş	Induced
Örüntü Tanıma	Pattern Recognation
Nitelikli Çizgeler	Atrributed Graphs
Veri Kümesi	Dataset
Yaklaşık Eşleme	Approximate Matching
Yeniden Oluşturma Listesi	Reconstruction List