

**KENAR AKTİF AĞ OLUŞTURMADA YAPAY KENAR
SAYISINI AZALTMAK İÇİN ETKİLİ BİR ALGORİTMANIN
TASARIM VE GERÇEKLEŞTİRİMİ : ALGORİTMANIN
LİTERATÜRDEKİ DİĞER ALGORİTMALAR İLE
KARŞILAŞTIRILMASI**

**DESIGN AND IMPLEMENTATION OF AN EFFICIENT
ALGORITHM TO DECREASE THE NUMBER OF DUMMY
ARCS IN AOA NETWORK : ITS COMPARISON WITH
ALGORITHMS IN THE LITERATURE**

SEVİL ŞEN

Hacettepe Üniversitesi

Fen Bilimleri Enstitüsü Yönetmeliğinin

BİLGİSAYAR MÜHENDİSLİĞİ Anabilim Dalı İçin Öngördüğü

YÜKSEK LİSANS (YÜKSEK MÜHENDİSLİK) TEZİ

olarak hazırlanmıştır

2005

Fen Bilimleri Enstitüsü Müdürlüğü'ne,

Bu çalışma jürimiz tarafından **BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI'nda YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Başkan

.....
Yrd. Doç. Dr. Kayhan İmre

Üye

.....
Yrd. Doç. Dr. Kıvanç Dinçer

Üye (Danışman)

.....
Yrd. Doç. Dr. Mustafa Ege

ONAY

Bu tez/...../..... tarihinde Enstitü Yönetim Kurulunca belirlenen yukarıdaki jüri üyeleri tarafından kabul edilmiştir.

...../...../.....

Prof. Dr. Ahmet R. ÖZDURAL
FEN BİLİMLERİ ENSTİTÜSÜ MÜDÜRÜ

KENAR AKTİF AĞ OLUŞTURMADA YAPAY KENAR SAYISINI AZALTMAK İÇİN ETKİLİ BİR ALGORİTMANIN TASARIM VE GERÇEKLEŞTİRİMİ : ALGORİTMANIN LİTERATÜRDEKİ DİĞER ALGORİTMALAR İLE KARŞILAŞTIRILMASI

Sevil ŞEN

ÖZ

Ağ tabanlı proje yönetiminde kullanılan Kenar Aktif Ağda(KAA) aktiviteler arası öncelik ilişkilerini ifade etmek için yapay kenarlar kullanılır. Az yapay kenar ile KAA'nın kurulması araştırmacıların ilgilendiği bir konu olmuştur. Literatürde bu problem, En Az Yapay Kenar Problemi olarak bilinmektedir. Problemin, NP'de tamamlanır problemler sınıfına girmesi nedeni ile, optimal çözüm için izlenebilir bir algoritma geliştirmek olası değildir, dolayısıyla, araştırmalar, olabildiğince az yapay kenarlı KAA'ı bulmaya yönelmiştir.

Bu tez çalışmasında, olabildiğince az sayıda yapay aktivite içeren kenar aktif ağın kurulması için etkili bir yaklaşım tasarlanmış ve gerçekleştirimi verilmiştir. Herhangi bir KAA üzerinde, aktiviteler arası öncelik ilişkilerini bozmadan yapay kenarların sayısını azaltacak bir indirgeme kurallar kümesi geliştirilmiştir. En kötü durumda polinomsal süreden daha fazla zaman gereksinimi duymayan ve küme kapsama problemi çözümünü de içermeyen indirgeme kuralları, dört ayrı sınıf altında incelenmiştir: Yapay kenar büzme, Yapay kenar ekleme, Düğüm örtüşme, Düğüm Ekleme. Geliştirilen indirgeme kurallarının işletim sırasının nasıl olması gerektiğine değinilmiş, indirgeme kurallarının farklı işleniş sırasının, hem işletim performansına hem de KAA oluşturmadaki etkisi araştırılmıştır. Ayrıca, önerilen yaklaşımın bilgisayar ortamında etkili gerçekleştirimi için kullanılması gereken veri yapıları açıklanmıştır.

Bu konuda son yıllarda yapılan çalışmalar incelenmiş ve tezdeki çalışma, söz konusu çalışmalar ile değişik kriterlere göre karşılaştırılmış ve karşılaştırma sonuçları tezde verilmiştir.

Anahtar Sözcükler: kenar aktif ağ oluşturma, en az yapay kenar problemi, polinomsal indirgeme kuralları, yapay kenar büzme, düğüm örtüşme

Danışman: Yrd. Doç. Dr. Mustafa EGE, Hacettepe Üniversitesi, Bilgisayar Mühendisliği Bölümü

DESIGN AND IMPLEMENTATION OF AN EFFICIENT ALGORITHM TO DECREASE THE NUMBER OF DUMMY ARCS IN AOA NETWORK : ITS COMPARISON WITH ALGORITHMS IN THE LITERATURE

Sevil ŞEN

ABSTRACT

In activity-on-arc(AoA) network which is just another diagram used in network-based project scheduling, dummy arcs are used to represent the immediate precedence relations between activities. AoA network construction with less number of dummy arcs is a research subject on which many researchers have studied. This problem is known as Minimum Dummy-Arc Problem. As it is classified as NP-complete, there doesn't exist a polynomial time algorithm, in other words, a tractable algorithm for the optimal solution can't be developed; therefore, researchers tend to handle the AoA network with the number of dummy arcs as minimum as possible.

In this thesis, an effective approach for constructing an AoA network including the number of dummy arcs as minimum as possible is designed and its implementation is given. We revise and improve the set of reduction rules described in order to decrease the number of dummy arcs without violating the precedence relations given in an AoA network. The improved reduction rules, whose worst case running time is polynomial, don't contain a solution for set cover problem and are examined under four different classifications: Dummy arc contraction, Dummy arc insertion, Node overlapping, Node insertion. We underline the importance of the processing order of the improved reduction rules and evaluate the effects of a different order of processing of the improved reduction rules on both the running time and AoA network construction. The necessary data structures for the effective implementation of our approach are also explained.

Recent studies given in literature are examined in view of various criteria, and our approach is compared to these studies and comparison results are also given.

Keywords: AoA network construction, minimum dummy arc problem, polynomial reduction rules, dummy arc contraction, node overlapping

Advisor: Asst. Prof. Dr. Mustafa EGE, Hacettepe University, Department of Computer Engineering.

TEŞEKKÜR

Yrd. Doç. Dr. Mustafa Ege'ye (tez danışmanı), tezin her aşamasında yol gösterdiği ve destek verdiğinden dolayı çok teşekkür ederim.

Tez jürisinde yer alan Yrd. Doç. Dr. Kayhan İmre ve Yrd. Doç. Dr. Kıvanç Dinçer'e, tezin son biçimini almasındaki katkılarından dolayı teşekkür ederim.

Manevi desteğini her zaman hissettiğim aileme teşekkür ederim.

İÇİNDEKİLER DİZİNİ

Sayfa

ÖZ	i
ABSTRACT	ii
TEŞEKKÜR	iii
İÇİNDEKİLER DİZİNİ	iv
ŞEKİLLER DİZİNİ	vi
ÇİZELGELER DİZİNİ	viii
1. GİRİŞ	1
1.1 Temel Kavramlar	4
1.1.1 Paralel Kenarlar	4
1.1.2 Geçişlilik İlişkisi	5
1.1.3 Fazlalık Aktiviteler	5
1.1.4 Küme Kapsama Problemi	6
2. KENAR AKTİF AĞ OLUŞTURMAK İÇİN SON YILLARDA YAPILAN	7
ÇALIŞMALAR	7
2.1 Ünlü'nün Yaklaşımı	7
2.1.1 Ara Çizgenin Oluşturulması	7
2.1.2 Kenar Aktif Ağın Oluşturulması	10
2.1.3 Algoritma Analizi	13
2.2 Güngör'ün Yaklaşımı	13
2.3 Michael, Kamburowski ve Stallman'ın Yaklaşımı	13
2.3.1 En Az Sayıda Düğüm İçeren Kenar Aktif Çatının Kurulması	14
2.3.2 Küme Kapsama Problemine İndirgeme	15
2.3.3 Algoritma Analizi	18
2.4 Sharma'nın Yaklaşımı	23
2.4.1 Fazlalık Aktiviteler	24
2.4.2 Algoritma Analizi	24
3. KENAR AKTİF AĞ OLUŞTURMAK İÇİN ÖNERİLEN YAKLAŞIM	25
3.1 İndirgeme Kuralları	25
3.1.1 Yapay Kenar Büzme Kuralları	26
3.1.1.1 1. Yapay Kenar Büzme Kuralı	26
3.1.1.2 2. Yapay Kenar Büzme Kuralı	27
3.1.1.3 3. Yapay Kenar Büzme Kuralı	29
3.1.2 Düğüm Örtüşme Kuralları	30
3.1.2.1 1. Düğüm Örtüşme Kuralı	30
3.1.2.2 2. Düğüm Örtüşme Kuralı	31
3.1.3 Yapay Kenar Ekleme Kuralları	33
3.1.3.1 1. Yapay Kenar Ekleme Kuralı	33
3.1.3.2 2. Yapay Kenar Ekleme Kuralı	34
3.1.3.3 3. Yapay Kenar Ekleme Kuralı	35
3.1.4 Düğüm Ekleme Kuralları	37
3.1.4.1 1. Düğüm Ekleme Kuralı	37
3.1.4.2 2. Düğüm Ekleme Kuralı	38
3.2 Önerilen Yaklaşımın Etkili Gerçekleştirimi	40
3.2.1 İndirgeme Kurallarının Ele Alınması	40
3.2.1.1 İndirgeme Listeleri	40
3.2.1.2 İndirgeme Kurallarının İşletim Sırası	41

3.2.1.3 Önerilen Algoritma	44
3.2.2 Veri Yapıları	58
3.2.2.1 Ara Çizge Veri Yapısı	58
3.2.2.2 Kenar Aktif Ağ Veri Yapısı	61
3.2.2.3 Paralel Kenar Veri Yapısı	67
3.2.2.3.1 Paralel Kenar Sayısı Veri Yapısı	67
3.2.2.3.2 Paralel Kenar Veri Yapısı	70
3.2.2.4 İndirgeme Veri Yapısı	71
3.3 Algoritma Analizi	74
4. ÖNERİLEN YAKLAŞIMIN DİĞER YAKLAŞIMLAR İLE KARŞILAŞTIRILMASI	75
4.1 Yakın Öncel İlişkilerin Bozulması Sorunu	79
5. SONUÇ VE ÖNERİLER	81
KAYNAKLAR DİZİNİ:	83
EK 1. SİMGELER VE KISALTMALAR DİZİNİ	85
EK 2. SÖZLÜK	85

ŞEKİLLER DİZİNİ

Sayfa

Şekil 1.1 Proje girdisi(a), bu projeye karşılık gelen DAA(b) ve iki farklı KAA gösterimi(c,d)	4
Şekil 1.2 Paralel kenarlar içeren bir KAA	4
Şekil 1.3 Geçişlilik ilişkisi içeren bir DAA	5
Şekil 1.4 Geçişlilik ilişkisi içeren bir KAA	5
Şekil 1.5 Fazlalık aktivite içeren bir KAA	5
Şekil 2.1 Bir DAA(a), bu ağa karşılık gelen öncel(b) ve ardıl tabanlı(c) proje girdileri	7
Şekil 2.2 Ara çizgenin oluşturulmasına ilişkin yordam[7]	9
Şekil 2.3 Proje girdisi(a) ve bu girdiye karşılık gelen ara çizge(b)	10
Şekil 2.4 Kenar aktif ağı oluşturulmasına ilişkin yordam[7]	12
Şekil 2.5 Şekil 2.3(a)'daki proje girdisinden yararlanılarak oluşturulan KAA	13
Şekil 2.6 [20] no'lu çalışmadan alınan bir DAA örneği(a) ve bu ağa karşılık gelen kenar aktif çatı(b)	14
Şekil 2.7 Şekil 2.6(a)'da yer alan DAA'a karşılık gelen KAA	18
Şekil 2.8 DAA örneği(a) ve bu ağa karşılık gelen kenar aktif çatı(b)	19
Şekil 2.9 Şekil 2.8(a)'da yer alan DAA'a karşılık gelen KAA	23
Şekil 2.10 Bir proje girdisi(a) ve bu proje girdisi için Sharma'nın algoritmasının ürettiği KAA(b)	24
Şekil 3.1 [21] no'lu çalışmadan alınan bir çizge örneği(a) ve bu çizgedeki $\langle e, j \rangle$ yapay kenarının 1. büzme kuralına göre büzülmesi sonucu oluşan çizge(b)	27
Şekil 3.2 Şekil 3.1(b)'de yer alan çizgedeki $\langle g, j \rangle$ yapay kenarının 2. büzme kuralına göre büzülmesi sonucu oluşan çizge	28
Şekil 3.3 [11] no'lu çalışmadan alınan bir çizge örneği(a) ve bu çizgedeki $\langle d, h \rangle$ yapay kenarının 3. büzme kuralına göre büzülmesi sonucu oluşan çizge(b)	29
Şekil 3.4 [11] no'lu çalışmadan alınan bir çizge örneği(a) ve bu çizgedeki b ve c düğümleri üzerinde 1. örtüşme kuralının uygulanması sonucu oluşan çizge(b)	31
Şekil 3.5 [11] no'lu çalışmadan alınan bir çizge örneği(a) ve bu çizgedeki b ve c düğümleri üzerinde 2. örtüşme kuralının uygulanması sonucu oluşan çizge(b)	32
Şekil 3.6 [20] no'lu çalışmadan alınan bir çizge örneği(a) ve bu çizgedeki a ve c düğümleri arasında 1. yapay kenar ekleme kuralının uygulanması sonucu oluşan çizge(b)	34
Şekil 3.7 [4] no'lu çalışmadan alınan bir çizge örneği(a) ve bu çizgedeki d ve e düğümleri arasında 2. yapay kenar ekleme kuralının uygulanması sonucu oluşan çizge(b)	35
Şekil 3.8 Bir çizge örneği(a) ve bu çizgedeki b ve e düğümleri arasında 3. yapay kenar ekleme kuralının uygulanması sonucu oluşan çizge(b)	36
Şekil 3.9 [20] no'lu çalışmadan alınan bir çizge örneği(a) ve bu çizgedeki b ve c düğümleri üzerinde 1. düğüm ekleme kuralının uygulanması sonucu oluşan çizge(b)	38
Şekil 3.10 Bir çizge örneği(a) ve bu çizgedeki g ve h düğümleri üzerinde 2. düğüm ekleme kuralının uygulanması sonucu oluşan çizge(b)	39
Şekil 3.11 [6] no'lu çalışmadan alınan örnek bir KAA(a) ve bu ağ üzerinde indirgeme kurallarının önerilen sırada(b) ve farklı sırada(c) uygulanması sonucu oluşan ağlar	42
Şekil 3.12 [17] no'lu çalışmadan alınan örnek bir KAA(a) ve bu ağ üzerinde indirgeme kurallarının önerilen sırada uygulanması sonucu elde edilen KAA(b)	43
Şekil 3.13 [17] no'lu çalışmadan alınan örnek bir KAA(a) ve indirgeme kurallarının farklı sırada işletilmesi sonucu oluşan ağlar(b,c)	43
Şekil 3.14 Kuralların, farklı çizgeler üzerinde, önerilen ve farklı sırada uygulanmasının işletilen kural sayısına etkisi(a) ve üretilen (x,y) ikili sayısına etkisi(b)	44
Şekil 3.15 İndirgeme kurallarının etkili gerçekleştirilmesi için önerilen algoritmanın ana yordamı	46
Şekil 3.16 İndirgeme kurallarına ilişkin yordamlara parametre olarak gönderilen (x,y) düğüm ikililerini üreten $x_y_Üret$ Yordamları	49
Şekil 3.17 Örnek bir KAA(a) ve bu ağ üzerinde indirgeme kuralları uygulanmadan önce V, E, D , indirgeme listeleri(b)	50
Şekil 3.18 3.17(a)'daki ağ üzerinde indirgeme kuralları uygulandıktan sonra KAA(a) ve V, E, D , indirgeme listeleri(b)	50
Şekil 3.19 1. Düğüm Örtüşme Kuralına ilişkin yordam	51
Şekil 3.20 2. Düğüm Örtüşme Kuralına ilişkin yordam	51
Şekil 3.21 1. Yapay Kenar Büzme Kuralına ilişkin yordam	52
Şekil 3.22 2. Yapay Kenar Büzme Kuralına ilişkin yordam	53
Şekil 3.23 3. Yapay Kenar Büzme Kuralına ilişkin yordam	53
Şekil 3.24 1. Yapay Kenar Ekleme Kuralına ilişkin yordam	54
Şekil 3.25 2. Yapay Kenar Ekleme Kuralına ilişkin yordam	54
Şekil 3.26 3. Yapay Kenar Ekleme Kuralına ilişkin yordam	55
Şekil 3.27 1. Düğüm Ekleme Kuralına ilişkin yordam	56
Şekil 3.28 2. Düğüm Ekleme Kuralına ilişkin yordam	57
Şekil 3.29 Ara Çizge Veri Yapısı	58

Şekil 3.30 Şekil 2.1.(b)'de yer alan proje girdisi Generate_Intermediary_Graph yordamında işlenirken, oluşturulan ara çizge veri yapısı ve ara çizgenin değişimi	60
Şekil 3.31 KAA'daki bir a düğümünün veri yapısı.....	61
Şekil 3.32 Bir DAA'ın KAA'a dönüşümü	62
Şekil 3.33 Şekil 3.30(b)'deki ara çizgeye karşılık gelen KAA veri yapısı	64
Şekil 3.34 Şekil 3.30(c)'de yer alan ağa karşılık gelen KAA veri yapısı	65
Şekil 3.35 Şekil 3.30(d)'de yer alan ağa karşılık gelen KAA veri yapısı	66
Şekil 3.36 KAA'daki bir d düğümü için paralel kenar sayısı veri yapısı	67
Şekil 3.37 Proje girdisi(a), bu proje girdisine karşılık gelen ara çizge(b) ve KAA(c), paralel kenarların ayıklanması sonucu elde edilen KAA(d)	68
Şekil 3.38 Şekil 3.37(b)'de yer alan ara çizge için paralel kenar sayısı veri yapısı(a) ve Şekil 3.37(c)'de yer alan KAA için paralel kenar sayısı veri yapısı(b).....	69
Şekil 3.39 KAA'daki bir d düğümü için paralel kenar veri yapısı	70
Şekil 3.40 İndirgeme veri yapısı	71
Şekil 3.41 [20] no'lu çalışmadan alınan bir KAA örneği ve bu ağa karşılık gelen indirgeme veri yapısı	73
Şekil 4.1 [14] no'lu çalışmadan alınan bir proje girdisi(a), DAA(b) ve bu proje girdisi için önerilen ve Michael ve diğ.'nin yaklaşımı ile elde edilen KAA(c)	79
Şekil 4.2 Şekil 4.1(b)'de verilen DAA'daki yakın öncellerin korunduğu KAA	80

ÇİZELGELER DİZİNİ

Sayfa

Çizelge 2.1 Şekil 2.6(a)'da yer alan DAA'daki her bir v düğümü için $S(v)$, $P^*(v)$, $P^*(s_v)$ ve $P^*(t_v)$ kümeleri	15
Çizelge 2.2 Şekil 2.6(b)'da yer alan kenar aktif çatıdaki her bir k düğümü için $P^*(k)$ kümesi ve k düğümüne ilişkin aktivitelerin uç noktaları	15
Çizelge 2.3 R kümesindeki her bir (i, j) ikilisi için $X(i, j)$, $Y(i, j)$, $X(i, j) \cap Y(i, j)$ ve $X(i, j) \cup Y(i, j)$ kümeleri	17
Çizelge 2.4 Şekil 2.8(a)'da yer alan DAA'daki her bir v düğümü için $S(v)$, $P^*(v)$, $P^*(s_v)$ ve $P^*(t_v)$ kümeleri	20
Çizelge 2.5 Şekil 2.8(b)'da yer alan kenar aktif çatıdaki her bir k düğümü için $P^*(k)$ kümesi ve k düğümüne ilişkin aktivitelerin uç noktaları	20
Çizelge 2.6 R kümesindeki her bir (i, j) ikilisi için $X(i, j)$, $Y(i, j)$, $X(i, j) \cap Y(i, j)$ ve $X(i, j) \cup Y(i, j)$ kümeleri	21
Çizelge 4.1 Kenar aktif ağ oluşturmada son yıllarda yapılmış olan çalışmaların karşılaştırılması	75
Çizelge 4.2 Literatürden alınan DAA örnekleri için, incelenen çalışmalar tarafından oluşturulan KAA'ların karşılaştırılması	76

1. GİRİŞ

Ağ tabanlı proje yönetiminde aktiviteler arası öncelik ilişkilerini ifade etmede genelde iki farklı gösterim tercih edilir: Düğüm Aktif Ağ(DAA)¹ ve Kenar Aktif Ağ(KAA)². Bir projeyi temsil eden tek bir düğüm aktif ağ gösterimi var iken, sınırsız sayıda kenar aktif ağ gösterimi olabilir. Ağ üzerinde uygulanan algoritmaların etkinliği, ağdaki yapay kenar ve düğüm sayısına bağlı olduğundan, en az sayıda düğüm ve yapay kenar içeren ağın kurulması önemlidir. Literatürde bu problem, *En Az Yapay Kenar Problemi* (Minimum Dummy Arc Problem) olarak bilinmektedir.

En Az Yapay Kenar Problemi, araştırmacıların[6,9,4,20,19,17,14,18,2,12,21,11] ilgilendiği bir konu olmuştur. Konu üzerinde ilk çalışmalar[6,9,4], en az sayıda yapay kenarı içeren KAA'ı bulmaya yöneliktir. Problemin, *NP'de tamamlanır*(NP-Complete) olduğu kanıtlandıktan[13] sonra, çalışmalar[10,18,12,21,7,8,11] olabildiğince az sayıda yapay kenar içeren KAA'ı veren yaklaşımlara yönelmiştir.

Problem üzerinde ilk çalışma, Dimsdale[6] tarafından yapılmıştır. Fisher ve diğ.[9] ve Syslo[20], Dimsdale'in algoritmasının yapay kenar ve düğüm sayılarını en aza indiremediği konusunda birer karşıt örnek vermişlerdir. Fisher ve diğ., Dimsdale'in algoritmasının fazladan yapay kenarlar ürettiğini de göstermiştir. Corneil ve diğ.[4] ve Syslo, Fisher'in algoritmasının yapay kenar sayısı en az olan KAA'ı oluşturamadığını ifade etmişlerdir.

Corneil ve diğ.[4], bir KAA, paralel kenarlar içermediğinde, KAA'da bulunan düğüm ve yapay kenarların eşzamanlı olarak en aza indirgenebileceğini savunsalar da, Syslo[20] ve Mrozek[17], bu savın genelde doğru olmadığını karşıt örnek vererek kanıtlamışlardır. Ayrıca, Corneil ve diğ., geliştirdikleri algoritmanın *küme kapsama*(set cover) problemi içermesi nedeni ile algoritma işletim süresinin kenar sayısı ile orantılı olarak üssel artabileceğini belirtmişlerdir.

En Az Yapay Kenar Problemi'nin *NP'de tamamlanır* olduğu ispatladıktan sonra, Syslo, "DAA'ın KAA'a dönüştürülmesinde yapay kenara gerek var mıdır?" sorusuna ilişkin yanıtın, polinomsal zamanda verilebileceğini göstermiştir.

¹Literatürde, aktivitelerin düğümler üzerinde olduğu ağa karşılık "Activity-on-Node (AoN)" ifadesi kullanılır.

²Literatürde, aktivitelerin kenarlar üzerinde olduğu ağa karşılık "Activity-on-Arc(AoA)" ya da "event-node network" ifadesi kullanılır.

Spinrad[19], bir DAA, bazı alt çizgeleri içermediğinde sorunu çözen sezgisel bir yaklaşım önerir. Güngör[11] ise, Sharma'nın[18] yaklaşımının ürettiği KAA'da, artık yapay kenarların bulunduğunu ifade etmiştir.

Michael ve diğ.[14], en az sayıda düğüm ile kenar aktif çatıyı(AoA framework) oluşturduktan sonra yapay kenarların sayısını azaltmaya çalışırlar. Fakat, Kamburowski[12], kenar aktif çatı en az sayıda düğüm ile kurulsa da *En Az Yapay Kenar Problemi*'nin, *NP'de tamamlanır* olarak kalmaya devam edeceğini vurgular.

Sorunun, *NP'de tamamlanır* problem sınıfına girmesi, en iyi çözümü veren *izlenebilir algoritmaların*(tractable algorithm) gerçekleştirmesini engelleyen en önemli etkidir. Ayrıca, KAA'ı elde etmek için doğrudan en iyi ya da en iyiye yakın çözümlere odaklaşılması, algoritmayı küme kapsama sorunu ile karşı karşıya bırakır. Bazı çözüm önerilerinin kısıtlanmış DAA'lar üzerinde çalışması, literatürde kabul görmüş bir test verisinin olmaması, yaklaşımların algoritma analizlerindeki eksiklik, çözüm önerilerinin karşılaştırılması konusunda engel oluşturmaktadır.

Araştırmacılar, genelde diğer yaklaşım(a/lara) kıyas ile daha az sayıda yapay kenar kullanarak KAA'ı elde ettiklerini ifade ederler. Bu tür değerlendirmeler için literatürde verilen örnek KAA'lar, "daha fazla yapay kenar içeren bir ağdan daha az yapay kenar içeren ağa dönüşüm belli kurallar çerçevesinde olası mıdır?" sorusu ile bizleri karşı karşıya bırakır.

Bu tez çalışmasında, olabildiğince az sayıda yapay aktivite içeren kenar aktif ağın kurulması için etkili bir yaklaşım tasarlanmış ve gerçekleştirimi verilmiştir. Düğüm aktif bir ağ, $G=(V,E)$ verildiğinde ($|V|=n$), işletim zamanı $\Omega(n)$ ile $O(n^2)$ arasında değişen bir ön işleme algoritması ile KAA'ı iki aşamada³ elde eden çalışmalardan[21,7] yararlanılmıştır. Ön işleme algoritmasının amacı, yapay kenar sayısını azaltmak değildir. Herhangi bir KAA üzerinde, aktiviteler arası öncelik ilişkilerini bozmadan yapay kenarların sayısını azaltacak bir indirgeme kuralları kümesi Güngör[11] tarafından önerilmiştir. Bu çalışmada, indirgeme kuralları tekrar gözden geçirilmiş ve geliştirilmiştir [8]. En kötü durumda polinomsal süreden daha fazla zaman gereksinimi duymayan ve *küme kapsama* problemi çözümünü de içermeyen indirgeme kuralları, dört ayrı sınıf altında incelenmiştir;

³ Aşamalardan birincisi, DAA'ın ara çizgeye(intermediary graph) çevrilmesidir. Diğeri ise, ara çizgenin proje kısıtlayıcılarını sağlayan bir KAA'a dönüştürülmesidir.

Yapay kenar bzme(Dummy arc contraction), *Yapay kenar ekleme*(Dummy arc insertion), *Dğm rtşme*(Node overlapping), *Dğm Ekleme*(Node insertion). nerilen yaklaşımin etkili gerekleştirmesi iin kullanılan veri yapıları aıklanmıştır. İşletim performansı aısından indirgeme kurallarının işleniş sırasının önemi vurgulanmıştır.

Tezin 2. bölümünde, bu konuda son yıllarda yapılan alışmalar incelenmiştir. İlk olarak, dğm aktif ađdan kenar aktif ađa dnşm en kt durumda $O(n^2)$ 'de gerekleştiren nl[21]'nn yaklaşımı incelenmiştir. İkinci olarak, kenar aktif ađdaki yapay kenarları azaltmak iin bzme kuralları tanımlayan Gngr[11]'n yaklaşımına değinilmiştir. Son olarak, en az yapay kenar problemini *kme kapsama* problemine indirgeyen Michael ve diğ.[14]'nin yaklaşımı ve yerel olarak bzme işlemini gerekleştiren Sharma[18]'nin yaklaşımı incelenmiştir.

Tezin 3. bölümünde, nerilen yaklaşım aıklanmıştır. Gngr[11] tarafından nerilen ve bu tez alışmasında geliştirilen indirgeme kuralları ayrıntılı bir şekilde aıklanmıştır. nerilen yaklaşımın bilgisayar ortamında etkili gerekleştirmesi iin indirgeme kurallarının nasıl ele alınacağı ve kullanılacak veri yapıları aıklanmıştır.

Tezin 4. bölümünde, nerilen yaklaşım; son yıllarda yapılan Sharma[18], Michael ve diğ.[14]'nin yaklaşımları ile deđişik aılardan karşılaştırılmıştır. Literatrden alınan dğm aktif ađ rnekleri iin, nerilen yaklaşım ve diğeri yaklaşımlar tarafından oluşturulmuş kenar aktif ađlar verilmiş ve bu ađlar yapay kenar, dğm sayıları aısından karşılaştırılmıştır.

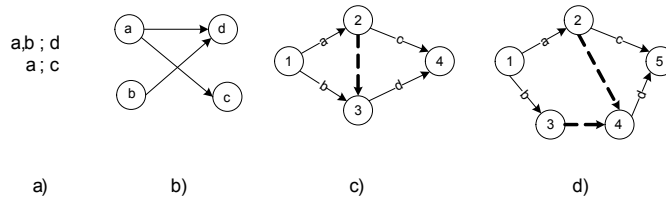
1.1 Temel Kavramlar

Büyük ölçekli projelerin programlanması, izlenmesi ve değerlendirilmesi için *Proje Değerlendirme ve Gözden Geçirme Tekniği*(PERT) ve *Kritik Yol Yöntemi*(CPM) geliştirilmiştir. Her iki yöntem de, bir projede yer alan aktiviteleri ve bu aktiviteler arasındaki öncel/ardıl ilişkilerini göstermek için düğüm aktif ya da kenar aktif gösterime dayalı ağları kullanırlar.

Düğüm Aktif Ağ(DAA) Gösterimi : DAA'larda, bir projedeki aktiviteler düğümler ile, aktiviteler arasındaki ilişkiler ise kenarlar ile ifade edilir. DAA, yapay kenar içermez. Bir projeyi temsil eden yalnızca bir DAA vardır.

Kenar Aktif Ağ(KAA) Gösterimi : KAA'larda, bir projedeki aktiviteler kenarlar ile, aktivitelerin bitiş ve başlangıçları ise düğümler ile ifade edilir. KAA, yapay kenarlar içerebilir. Bir projeyi temsil eden birçok KAA oluşturulabilir.

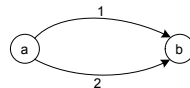
Şekil 1.1'de verilen bir proje girdisine karşılık gelen DAA ve iki farklı KAA gösterimi verilmiştir. Şekil 1.1(a)'da verilen proje girdisi, "d aktivitesinin başlayabilmesi için a ve b aktivitelerinin, c aktivitesinin başlayabilmesi için a aktivitesinin bitmesi gerektiği" bilgisini içerir. Şekil 1.1(c)'deki KAA'da, "a ve b den sonra d aktivitesi başlar" ilişkisini gösterebilmek için <2,3> yapay aktivitesi kullanılmıştır.



Şekil 1.1 Proje girdisi(a), bu projeye karşılık gelen DAA(b) ve iki farklı KAA gösterimi(c,d)

1.1.1 Paralel Kenarlar

Bir çizgedeki aynı kuyruk ve baş düğümüne sahip farklı kenarlara paralel kenarlar denir. Şekil 1.2'de, paralel kenarlar içeren bir KAA verilmiştir.

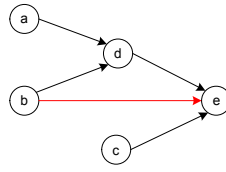


Şekil 1.2 Paralel kenarlar içeren bir KAA

1.1.2 Geçişlilik İlişkisi

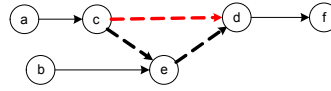
Bir çizgedeki bir x aktivitesinin öncellerinden herhangi birinin yakın öncelleri kümesi ile, x aktivitesinin yakın öncelleri kümesinin kesişimi boş küme değil ise geçişlilik ilişkisi ortaya çıkar[11].

Şekil 1.3'de geçişlilik ilişkisi içeren bir DAA verilmiştir. Ağdaki $\langle b,e \rangle$ kenarı; e aktivitesinin yakın öncelleri $\{b,c,d\}$ kümesi ile bu kümede yer alan d aktivitesinin yakın öncelleri $\{a,b\}$ kümesinin kesişimi $\{b\}$ boş küme olmadığından, geçişliliğe neden olur.



Şekil 1.3 Geçişlilik ilişkisi içeren bir DAA

Şekil 1.4'de ise $\langle c,d \rangle$ yapay kenarından dolayı geçişlilik ilişkisi içeren bir KAA verilmiştir.

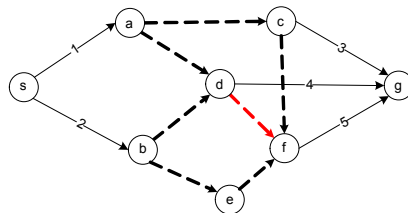


Şekil 1.4 Geçişlilik ilişkisi içeren bir KAA

1.1.3 Fazlalık Aktiviteler

Kenar aktif ağda yer alan ve ağdan çıkarılması ile ağın öncel/ardıl ilişkilerinde bir değişiklik yaratmayan aktivitelere fazlalık aktiviteler denir.

Şekil 1.5'de fazlalık aktivite içeren bir KAA verilmiştir. Ağdaki $\langle d,f \rangle$ yapay aktivitesinin çıkarılması ile ağdaki öncel/ardıl ilişkileri bozulmaz, dolayısıyla $\langle d,f \rangle$ kenarı fazlalık bir aktivitedir.



Şekil 1.5 Fazlalık aktivite içeren bir KAA

1.1.4 Küme Kapsama Problemi

X ve F kümeleri verilmiş olsun. F kümesi, sonlu X 'in alt kümelerini içeren bir küme olsun. X kümesinin her bir elemanı F kümesinin en az bir alt kümesinde yer alıyor ise $(X = \bigcup_{S \in F} S)$; $S \in F$ kümesi X kümesinin elemanlarını kapsar denir. Literatürde

X kümesinin tüm elemanlarını kapsayan en az elemanlı $C \subseteq F$ alt kümesini bulma problemine *küme kapsama* problemi denir : $X = \bigcup_{S \in C} S$

Küme kapsama problemi *NP*'de tamamlanır bir problemdir[16]. Bu nedenle, literatürde en az elemanlı C alt kümesini bulmak yerine, olabildiğince az elemanlı C alt kümesini veren sezgisel yaklaşımlar vardır.

2. KENAR AKTİF AĞ OLUŞTURMAK İÇİN SON YILLARDA YAPILAN ÇALIŞMALAR

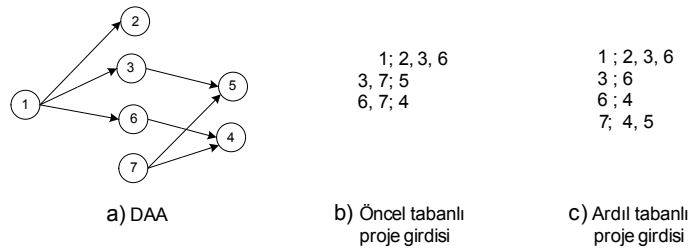
2.1 Ünlü'nün Yaklaşımı

Bu çalışmada, KAA oluşturmak için yeni bir yaklaşım önerilmiştir. Bu yaklaşımda, optimal çözümü veren bir KAA oluşturmak yerine, düğüm aktif ağı kenar aktif bir ağa dönüştüren etkili bir ön işleme algoritması verilmiştir. Ön işleme algoritmasının karmaşıklığı $\Omega(n)$ ile $O(n^2)$ arasındadır.

Ön işleme algoritması, iki aşamada gerçekleştirilir. Birinci aşamada, DAA ara çizgeye dönüştürülür. Ara çizge, proje girdisinde verilen öncelik ilişkilerini koruyan, fakat proje kısıtlayıcılarını sağlamayan bir kenar aktif çizgedir. İkinci aşamada ise ara çizgede bulunan, proje kısıtlayıcılarına uymayan durumlar düzeltilerek KAA oluşturulur.

2.1.1 Ara Çizgenin Oluşturulması

Ara çizgenin oluşturulmasına ilişkin yordam(Generate_Intermediary_Graph), bir DAA'daki öncelik ilişkilerini ele alan Şekil 2.1.(b) veya Şekil 2.1.(c)'deki gibi bir proje girdisini kullanır. Proje girdisi, ardıl tabanlı ya da öncel tabanlı olabilir. Proje girdisine göre bir DAA için iki farklı ara çizge elde edilebilir.



Şekil 2.1 Bir DAA(a), bu ağa karşılık gelen öncel(b) ve ardıl tabanlı(c) proje girdileri

Proje girdisindeki her bir satırda “;” den sonra yer alan aktivitelerin başlayabilmesi, aynı satırdaki “;” den önce yer alan aktivitelerin tamamlanmasına bağlıdır. Dolayısıyla ara çizgenin oluşturulmasına ilişkin yordam, “proje girdisinde ‘;’ den önce yer alan aktiviteler bir noktada buluşmalı ki ‘;’ den sonra yer alan aktiviler bu noktadan başlayabilsin” temel mantığına dayanır. Bir DAA, $G=(V,E)$ verildiğinde ara çizgenin oluşturulmasına ilişkin yordam (Generate_Intermediary_Graph) Şekil 2.2’de verilmiştir. Bu yordamda kullanılan semboller aşağıda verilmiştir :

N : Proje girdisindeki satır sayısı.

P_i : Proje girdisinin i . satırında ';' den önce yer alan aktiviteler kümesi.

S_i : Proje girdisinin i . satırında ';' den sonra yer alan aktiviteler kümesi.

p_i^j : i . satırdaki P_i kümesinin j . elemanı $1 \leq j \leq |P_i|$.

s_i^j : i . satırdaki S_i kümesinin j . elemanı $1 \leq j \leq |S_i|$.

Proje girdisi işlendikten sonra oluşacak ara çizge $G'=(V',E')$ ile temsil edilsin.

V' : Ara çizgede yer alan düğümler kümesidir, başlangıçta boş kümedir.

$$V' = \{v_1, v_2, \dots\}$$

E' : Ara çizgede yer alan kenarlar kümesidir, başlangıçta boş kümedir. E' kümesi üçlülerden oluşur ve aşağıdaki biçimde tanımlanır:

$$E' = \{ \langle tail, head, name \rangle : tail, head \in V', tail \neq head, name \in V \}$$

Ara çizge kurulurken henüz atanmamış kuyruk ve baş düğümler için '*' sembolü kullanılacaktır.

H_1^j : Kuyruk düğümü atansın atanmasın fakat bitiş düğümü atanmış ve adı j olan aktivitelerin bitiş düğümlerini tutan küme.

H_2^j : Kuyruk düğümü atanmamış ve adı j olan aktivitelerin bitiş düğümlerini tutan küme.

T_1^j : Bitiş düğümü atansın atanmasın fakat kuyruk düğümü atanmış ve adı j olan aktivitelerin kuyruk düğümlerini tutan küme.

T_2^j : Bitiş düğümü atanmamış ve adı j olan aktivitelerin kuyruk düğümlerini tutan küme. ($1 \leq j \leq |V|$).

Generate_Intermediary_Graph()

```

1. k := 'a'; V' := {}; E' := {};
2. for (i := 1 to N){
3.   V := V' ∪ {k}
4.   w := k;
5.   for (j := 1 to |Pi|) {
6.     if (∀tail,head [<tail,head, pij> ∉ E' ∧ tail, head ∈ V' ∪ {*}])
7.       E' := E' ∪ {<*, w, pij>};
8.     else
9.       if (¬{∃tail [<tail, *, pij> ∈ E' ∧ tail ∈ V']})
10.        E' := E' ∪ {<*, w, pij>};
11.      else
12.        t := min{tail | <tail, *, pij> ∈ E' for each tail ∈ V'}
13.        <t, *, pij> := <t, w, pij>;
14.    }
15.   for (j := 1 to |Si|){
16.     if (∀tail,head [<tail,head, Sij> ∉ E' ∧ tail, head ∈ V' ∪ {*}])
17.       E' := E' ∪ {<w, *, Sij>};
18.     else
19.       if (¬{∃head [<*, head, Sij> ∈ E' ∧ head ∈ V']})
20.        E' := E' ∪ {<w, *, Sij>};
21.      else
22.        h := min{head | <*, head, Sij> ∈ E' for each head ∈ V'}
23.        <*, h, Sij> := <w, h, Sij>;
24.    }
25.   k := next(k);
26. }
27. start := k; V' := V' ∪ {start};
28. finish := next(k); V' := V' ∪ {finish};
29. for (j := 1 to |V|){
30.   if (∃head [<*, head, j> ∈ E' ∧ head ∈ V']){
31.     H1j := {head | <tail,head,j> ∈ E' for each tail ∈ V' ∪ {*}, head ∈ V'}
32.     if (∀head [<*, head, j> ∈ E' ∧ head ∈ H1j])
33.       t := start;
34.     else
35.       t := max{tail | <tail,head,j> ∈ E', tail, head ∈ V'};
36.     H2j := {head | <*,head,j> ∈ E' for each head ∈ V'}
37.     for (each head ∈ H2j)
38.       <*, head, j> := <t, head, j>;
39.   }
40.   if (∃tail [<tail, *, j> ∈ E' ∧ tail ∈ V']){
41.     T1j := {tail | <tail, head, j> ∈ E' for each tail ∈ V', head ∈ V' ∪ {*}};
42.     if (∀tail [<tail, *, j> ∈ E' ∧ tail ∈ T1j])
43.       h := finish;
44.     else
45.       h := max{head | <tail,head,j> ∈ E', tail, head ∈ V'};
46.     T2j := {tail | <tail, *, j> ∈ E' for each tail ∈ V'};
47.     for (each tail ∈ T2j)
48.       <tail, *, j> := <tail, h, j>;
49.   }
50. }

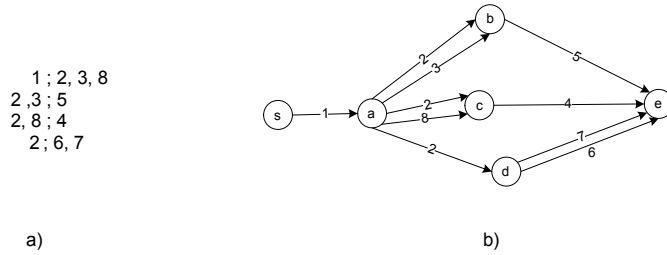
```

Şekil 2.2 Ara çizgenin oluşturulmasına ilişkin yordam[7]

Ara çizge, aşağıdaki proje kısıtlayıcılarını sağlamaz :

- Bir aktivite, KAA'da birden fazla kenar ile temsil edilemez.
- KAA, paralel kenarlar içermez.

Şekil 2.3'de, [21]'de verilen bir proje girdisi ve bu girdiye karşılık gelen ara çizge verilmiştir. Şekil 2.3(b)'deki ara çizgede, 2 no'lu aktivite birden fazla kenarlar ile temsil edilmektedir. Ayrıca, paralel kenarların görüldüğü bazı düğüm çiftlerini de($\{<a,b>, <a,c>, <d,e>\}$) içermektedir.



Şekil 2.3 Proje girdisi(a) ve bu girdiye karşılık gelen ara çizge(b)

2.1.2 Kenar Aktif Ağın Oluşturulması

Ara çizgenin KAA'a dönüşümü sırasında aşağıdaki hususlar göz önüne alınmaktadır :

- Proje girdisindeki öncelik ilişkileri korunmalıdır.
- Ara çizgede bulunan proje kısıtlayıcıları kaldırılmalıdır.
- KAA, yerel olarak olabildiğince az sayıda yapay aktivite ile oluşturulmaya çalışılmalıdır.

KAA'ın oluşturulmasına ilişkin yordamda(AoA_Network_Construction), öncelikle ara çizgede birden fazla kenar ile temsil edilen aktiviteler incelenir ve KAA'da her bir aktivitenin tek bir kenar ile temsil edilmesi sağlanır. Daha sonra, ara çizgede bulunan paralel aktiviteler yok edilir.

Ara çizgeden KAA'a geçişte oluşan çizge $G=(V, E)=(V' \cup V_D, E' \cup E_D)$ ile temsil edilsin. KAA'ın oluşturulmasına ilişkin yordamda kullanılan semboller aşağıda verilmiştir :

V' : KAA'da oluşacak gerçel düğümler kümesidir. Ara çizgedeki düğümler kümesine eşittir ($V' \leftarrow V$). Bu küme aşağıdaki biçimde tanımlanır:

$$V' = \{v_1, v_2, \dots\}$$

E' : KAA'da oluşacak gerçel kenarlar kümesidir. Başlangıçta, ara çizgedeki kenarlar kümesine eşittir ($E' \leftarrow E$). Dönüşüm sırasında bazı kenarlar E' kümesinden çıkartılabilir veya bazı kenarlar E' kümesine eklenebilir. Bu küme aşağıdaki biçimde tanımlanır:

$$E' = \{ \langle \text{tail}, \text{head}, \text{name} \rangle : \text{tail} \in (V' \cup V_D), \text{head} \in (V' \cup V_D), \text{name} \in V \}$$

V_D : KAA'da oluşacak yapay düğümler kümesidir. Başlangıçta boş kümedir ($V_D \leftarrow \{ \}$). Bu küme aşağıdaki biçimde tanımlanır:

$$V_D = \{d_1, d_2, \dots\}$$

E_D : KAA'da oluşacak yapay kenarlar kümesidir. Başlangıçta boş kümedir ($E_D \leftarrow \{ \}$). Bu küme aşağıdaki biçimde tanımlanır:

$$E_D = \{ \langle \text{tail}, \text{head}, \text{name} \rangle : \text{tail} \in (V' \cup V_D), \text{head} \in (V' \cup V_D), \text{name} \in \{-1, -2, \dots\} \}$$

Ara çizgede, bir x aktivitesinin birden fazla kenarla temsil edildiğini varsayalım:

T : x aktivitesinin kuyruk düğümlerinin kümesidir. $T: \{t_1, t_2, \dots\}$

H : x aktivitesinin bitiş düğümlerinin kümesidir. $H: \{h_1, h_2, \dots\}$

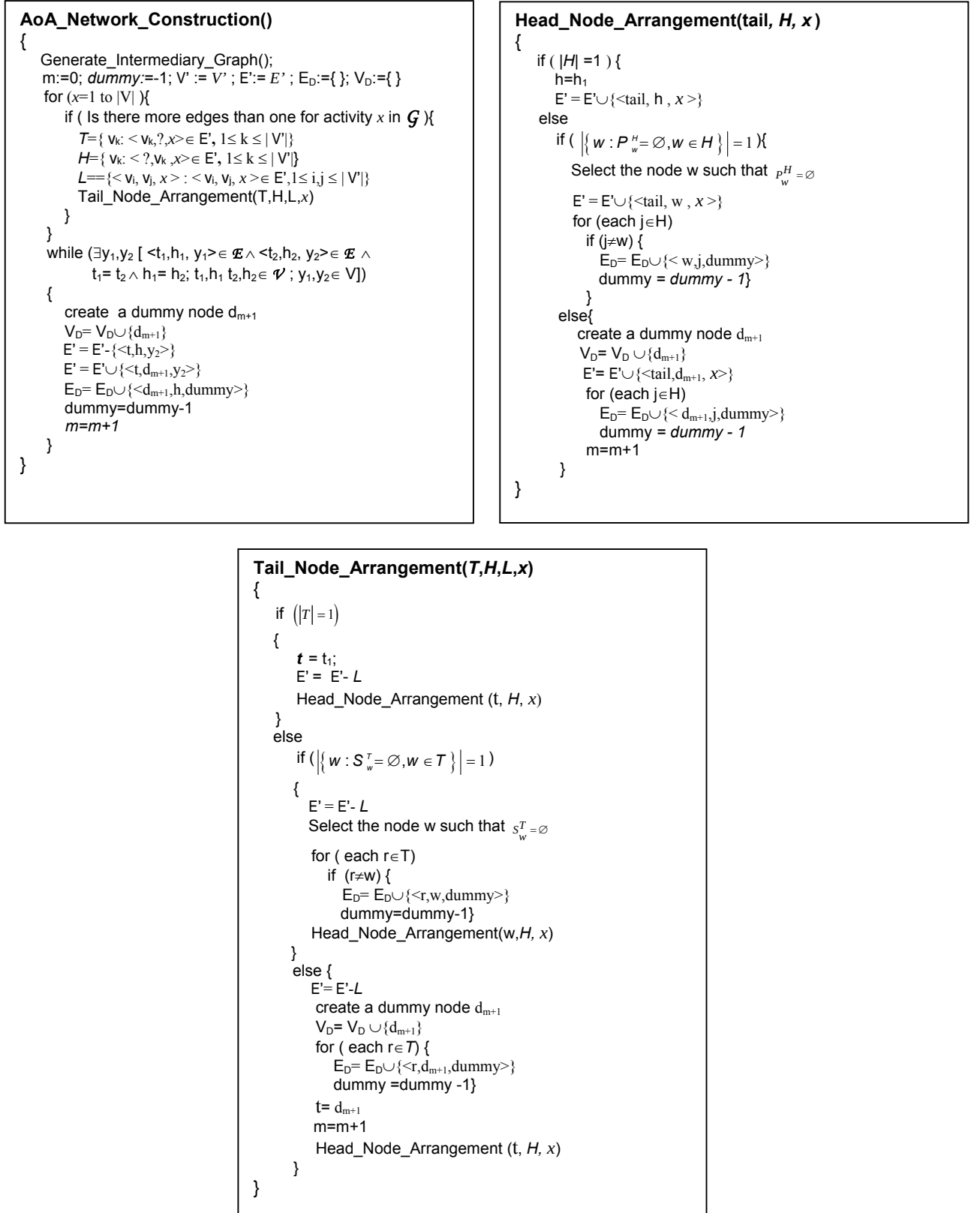
P_w^T : w ($w \in T$) düğümünde sonlanan aktivitelerin adlarını içerir. P_w^T , w düğümü için öncel kümedir.

S_w^T : w ($w \in T$) düğümünden başlayan aktivitelerin adlarını içerir. S_w^T , w düğümü için ardıl kümedir.

P_w^H : w ($w \in H$) düğümünde sonlanan aktivite adlarını içerir. P_w^H , w düğümü için öncel kümedir.

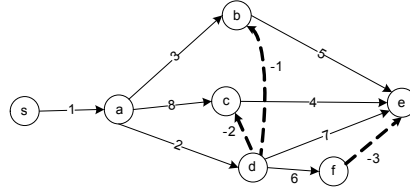
S_w^H : w ($w \in H$) düğümünden başlayan aktivitelerin adlarını içerir. S_w^H , w düğümü için ardıl kümedir.

Bir DAA'a ilişkin proje girdisinden yararlanarak, KAA'ı oluşturan yordam (AoA_Network_Construction) Şekil 2.4'de verilmiştir :



Şekil 2.4 Kenar aktif ağın oluşturulmasına ilişkin yordam[7]

Şekil 2.5'te, Şekil 2.3'teki proje girdisinden yararlanılarak oluşturulan KAA verilmiştir. Görüldüğü gibi KAA, Şekil 2.3(b)'deki proje kısıtlayıcılarını içermez.



Şekil 2.5 Şekil 2.3(a)'daki proje girdisinden yararlanılarak oluşturulan KAA

2.1.3 Algoritma Analizi

Ünlü'nün yaklaşımında verilen KAA'nın oluşturulmasına ilişkin yordamın algoritma karmaşıklığı $\Theta(|E'|)$ 'dir. Bir DAA'da ($G=(V,E)$) aktivite sayısı $|V|=n$ ise, ara çizgedeki maksimum kenar sayısının $|E'|=n^2$ olduğu ispatlanmıştır[7]. Dolayısıyla, Ünlü'nün yaklaşımında önerilen algoritma $O(n^2)$ ile üstten sınırlıdır.

2.2 Güngör'ün Yaklaşımı

Güngör'ün çalışmasında, herhangi bir KAA üzerinde aktiviteler arası öncelik ilişkilerini bozmadan yapay kenarların sayısını azaltmak için en kötü durumda polinomsal süreden daha fazla zaman gereksinimi duymayan ve *küme kapsama* problemi çözümünü de içermeyen bir büzme kurallar kümesi tanımlanmıştır. Güngör'ün önerdiği ve bu tez çalışmasında geliştirilen indirgeme kuralları Bölüm 3.1'de ayrıntılı olarak açıklanmıştır.

2.3 Michael, Kamburowski ve Stallman'ın Yaklaşımı

Bir DAA'a karşılık gelen en az sayıda düğüm içeren KAA'nın en az sayıda yapay aktivite ile kurulması *NP'de tamamlanır* bir problemdir. Bu çalışmada, Michael ve diğ., bu problemin polinomsal zamanda *küme kapsama* problemine indirgenebildiğini göstermişlerdir. Böylece bilinen *küme kapsama* algoritmaları, problemin çözülmesi için uygulanabilir.

Problemin çözümü iki temel aşamada gerçekleşir :

- en az sayıda düğüm içeren kenar aktif çatının kurulması
- problemin *küme kapsama* problemine indirgenmesi

2.3.1 En Az Sayıda Düğüm İçeren Kenar Aktif Çatının Kurulması

En az sayıda düğüm içeren kenar aktif çatının kurulma algoritması, ilk olarak Cantor ve Dimsdale[3] tarafından geliştirilmiştir. Bu algoritmanın temel mantığı aşağıda verilmiştir.

$G = (V, E)$ düğüm aktif ağı verilmiş olsun. $P^*(v)$ kümesi, v aktivitesinin tüm öncellerini; $S(v)$ kümesi ise v aktivitesinin yakın ardıllarını içerir. DAA'daki her bir v düğümüne karşılık gelen aktivite, kenar aktif ağda (s_v, t_v) düğüm ikilisi ile ifade edilsin. $P^*(s_v)$ ve $P^*(t_v)$ kümelerinin elde edilişi aşağıda verilmiştir :

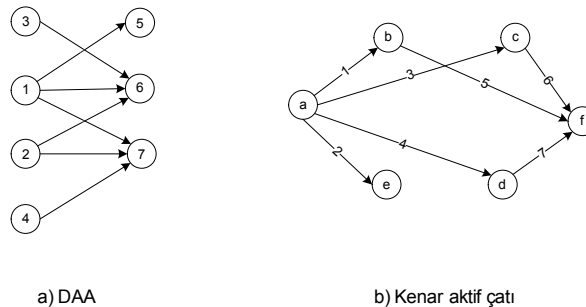
$$P^*(s_v) = P^*(v)$$

$$P^*(t_v) = \bigcap_{w \in S(v)} P^*(w) \text{ (Kesişim boş küme ise } P^*(t_v) = V \text{ 'dir)}$$

KAA'da yer alan düğümlerin kümesi $N = \{1, 2, \dots, n\}$ olsun.

$\{P^*(k) \mid k = \{1, 2, \dots, n\}\}$ kümesi, $\{P^*(s_v) \cup P^*(t_v)\}$ kümesinde yer alan farklı kümeleri içerir. $P^*(k)$, kenar aktif çatıda yer alan k düğümünün tüm öncellerini ifade eder. Her bir v aktivitesi için, bir (i, j) aktivitesi atanır ($P^*(i) = P^*(s_v)$ ve $S(j) = S^*(t_v)$). Aynı düğümü paylaşabilecek düğümler belirlendiğinden, N kümesinde en az sayıda düğüm vardır.

Şekil 2.6(a)'da; [20] numaralı çalışmadan alınan bir DAA yer almaktadır. Çizelge 2.1'de, bu DAA'daki her bir düğüm için $S(v)$, $P^*(v)$, $P^*(s_v)$ ve $P^*(t_v)$ kümeleri verilmiştir. Çizelge 2.2'de, KAA'da yer alacak olan her bir k düğümü için $P^*(k)$ kümesi ile bu düğümde başlayan ve biten aktivitelerin uç noktaları verilmiştir. Şekil 2.6(b)'de, bu DAA'a karşılık gelen kenar aktif çatı verilmiştir.



Şekil 2.6 [20] no'lu çalışmadan alınan bir DAA örneği(a) ve bu ağa karşılık gelen kenar aktif çatı(b)

Çizelge 2.1 Şekil 2.6(a)'da yer alan DAA'daki her bir v düğümü için $S(v)$, $P^*(v)$, $P^*(s_v)$ ve $P^*(t_v)$ kümeleri

v	$S(v)$	$P^*(v)$	$P^*(s_v)$	$P^*(t_v)$
1	5,6,7	-	-	1
2	6,7	-	-	1,2
3	6	-	-	1,2,3
4	7	-	-	1,2,4
5	-	1	1	1,2,3,4,5,6,7
6	-	1,2,3	1,2,3	1,2,3,4,5,6,7
7	-	1,2,4	1,2,4	1,2,3,4,5,6,7

Çizelge 2.2 Şekil 2.6(b)'da yer alan kenar aktif çatıdaki her bir k düğümü için $P^*(k)$ kümesi ve k düğümüne ilişkin aktivitelerin uç noktaları

k	$P^*(k)$	uç noktalar
a	-	s_1, s_2, s_3, s_4
b	1	s_5, t_1
c	1,2,3	s_6, t_3
d	1,2,4	s_7, t_4
e	1,2	t_2
f	1,2,3,4,5,6,7	t_5, t_6, t_7

2.3.2 Küme Kapsama Problemine İndirgeme

Şekil 2.6(b)'de görüldüğü üzere kenar aktif çatı, Şekil 2.6(a)'da yer alan DAA'daki tüm öncelik ilişkilerini içermez. Örneğin $\langle 1,6 \rangle$ ilişkisi DAA'da yer almasına rağmen, kenar aktif çatıda yer almaz. DAA'daki bir $\langle u,v \rangle$ ilişkisini göstermek için, $\pi(t_u, s_v)$ yapay yolu kenar aktif çatıya eklenmelidir. İki ya da daha fazla aktivite aynı baş ya da/ve kuyruk düğümlere sahip olabileceğinden, kenar aktif çatıda yer almayan her bir ilişki için çatıya bir yapay yol eklenmesi etkili bir yöntem değildir. Aşağıda, Michael ve diğ.'nin önerdiği yöntem açıklanmıştır[14] :

Öncelikle, kenar aktif çatıda yer almayan ilişkilerin yer aldığı bir R kümesi oluşturulur :

$$R = \{ (i, j) \mid i \neq j \text{ ve } \exists (u,v) \in G, t_u=i \text{ ve } s_v=j \}$$

R kümesindeki her bir (i, j) ikilisi için $X(i, j)$ ve $Y(i, j)$ kümeleri oluşturulur :

$$X(i, j) = \{k \mid (k, j) \in R \text{ ve } P^*(i) \subset P^*(k)\}$$

$$Y(i, j) = \{l \mid (i, l) \in R \text{ ve } P^*(l) \subset P^*(j)\}$$

R kümesi aşağıda verildiği gibi üç farklı kümeye bölünür :

$$R_1 = \{(i, j) \mid X(i, j) \cap Y(i, j) \neq \emptyset\}$$

$$R_2 = \{(i, j) \mid X(i, j) \cup Y(i, j) = \emptyset\}$$

$$R_3 = R - (R_1 \cup R_2)$$

R_1 kümesi, R kümesinde yer alan ikililerin bir alt kümesidir. R_1 kümesinde yer alan ikililer, kenar aktif çatıya bir yapay aktivite eklemeksizin sağlanan öncelik ilişkilerini gösterir. R_2 kümesinde yer alan (i, j) ikilileri, kenar aktif çatıya yalnızca (i, j) yapay aktivitesinin eklenmesi ile sağlanabilen öncelik ilişkilerini gösterir. R_3 kümesinde, R_1 ve R_2 kümesinde yer almayan ve yapay aktivite problemine neden olan ikililer yer alır. R_3 kümesindeki yer alan ilişkileri en az sayıda yapay aktivite ile sağlayabilmek için aşağıdaki kümeler tanımlanır :

$$F = \bigcup_{(i, j) \in R_3} F(i, j)$$

$$F(i, j) = F_1(i, j) \cup F_2(i, j) \cup F_3(i, j)$$

$$F_1(i, j) = \{(i, k) \mid k \in X(i, j)\}$$

$$F_2(i, j) = \{(l, j) \mid l \in Y(i, j)\}$$

$$F_3(i, j) = \{(l, k) \mid P^*(l) \subset P^*(k), k \in X(i, j), l \in Y(i, j)\}$$

$$C = \{Q_{pq} \mid (p, q) \in F\}$$

$$Q_{pq} = \{(i, j) \mid (i, j) \in R_3, (p, q) \in F(i, j)\}$$

$F(i, j)$ kümesi, R_3 kümesindeki (i, j) ilişkisini KAA'da göstermek için eklenmesi gereken yapay kenarları içerir. Q_{pq} kümesi, (i, j) ikililerini içerir. Eğer $F(i, j)$ kümesinde (p, q) ikilisi yer alıyor ise (i, j) ikilisi, Q_{pq} kümesine eklenir.

Şekil 2.6(b)'da verilen kenar aktif çatıda yer almayan ilişkiler ve bu ilişkilerin sağlanması için eklenmesi gereken $\langle i,j \rangle$ ikililerini içeren R kümesi aşağıda verilmiştir :

$$\text{İlişkiler} = \{\langle 1,6 \rangle, \langle 1,7 \rangle, \langle 2,6 \rangle, \langle 2,7 \rangle\}$$

$$R = \{\langle b,c \rangle, \langle b,d \rangle, \langle e,c \rangle, \langle e,d \rangle\}$$

Çizelge 2.3'te, R kümesindeki her bir (i, j) ikilisi için oluşturulan $X(i, j)$ ve $Y(i, j)$ kümeleri verilmiştir.

Çizelge 2.3 R kümesindeki her bir (i, j) ikilisi için $X(i, j)$, $Y(i, j)$, $X(i, j) \cap Y(i, j)$ ve $X(i, j) \cup Y(i, j)$ kümeleri

R	$X(i, j)$	$Y(i, j)$	$X(i, j) \cap Y(i, j)$	$X(i, j) \cup Y(i, j)$
$\langle b,c \rangle$	$\{e\}$	$\emptyset$	$\emptyset$	$\{e\}$
$\langle b,d \rangle$	$\{e\}$	$\emptyset$	$\emptyset$	$\{e\}$
$\langle e,c \rangle$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$\langle e,d \rangle$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

Çizelge 2.3'te verilen kümelere göre elde edilen R_1 , R_2 ve R_3 kümeleri aşağıda verilmiştir :

$$R_1 = \emptyset$$

$$R_2 = \{\langle e,c \rangle, \langle e,d \rangle\}$$

$$R_3 = \{\langle b,c \rangle, \langle b,d \rangle\}$$

R_3 kümesinde yer alan her bir $\langle i,j \rangle$ ikilisi için oluşturulan $F(i, j)$ kümeleri aşağıda verilmiştir.

$$F = \{ F(b, c), F(b, d) \}$$

$$F(b, c) = F_1(b, c) \cup F_2(b, c) \cup F_3(b, c) = \{\langle b,e \rangle \cup \emptyset \cup \emptyset\} = \{\langle b,e \rangle\}$$

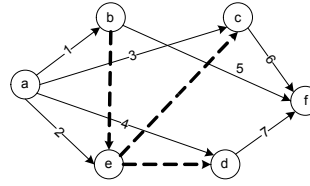
$$F(b, d) = F_1(b, d) \cup F_2(b, d) \cup F_3(b, d) = \{\langle b,e \rangle \cup \emptyset \cup \emptyset\} = \{\langle b,e \rangle\}$$

$$C = \{Q_{be}\}$$

$$Q_{be} = \{\langle b,c \rangle, \langle b,d \rangle\}$$

Problem, burada *küme kapsama* problemine indirgenir. En az sayıda Q_{pq} kümesinin birleşimi ile R_3 kümesi elde edilmeye çalışılır.

Yukarıda verilen örnekte C kümesinde sadece Q_{be} elemanı olduğundan, R_3 kümesindeki $\langle b,c \rangle$ ve $\langle b,d \rangle$ ilişkilerini sağlamak için $\langle b,e \rangle$ aktivitesinin eklenmesi yeterlidir. Sonuç olarak, kenar aktif çatıya eklenmesi gereken aktiviteler $\{R_2 \cup \langle b,e \rangle\}$ 'dir. Şekil 2.7'de, Şekil 2.6(a)'da verilen DAA'a karşılık gelen KAA verilmiştir.



Şekil 2.7 Şekil 2.6(a)'da yer alan DAA'a karşılık gelen KAA

2.3.3 Algoritma Analizi

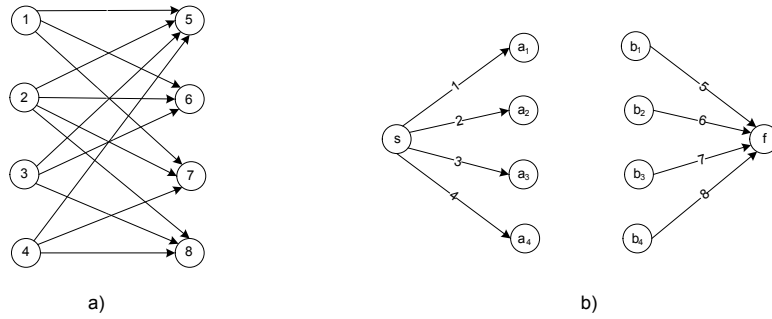
En az yapay kenar probleminin optimal çözümü için [14]'de verilen çalışmada önerilen algoritma aşağıda verilmiştir :

1. DAA'daki her bir v düğümü için oluşturulan $P^*(s_v)$ ve $P^*(t_v)$ kümelerinden farklı kümeleri bul.
2. DAA'daki her bir v aktivitesi için bir $\langle i,j \rangle$ ($P^*(i)=P^*(s_v)$ ve $S(j)=S^*(t_v)$) kenarı atayarak kenar aktif çatıyı kur.
3. $\langle i,j \rangle \mid i \neq j$ ve $\exists (u,v) \in G, t_u=i$ ve $s_v=j$ koşulunu sağlayan $\langle i,j \rangle$ ikililerini R kümesine ekle.
4. R kümesindeki her bir $\langle i,j \rangle$ ikilisi için $X(i,j)$ ve $Y(i,j)$ kümelerini oluştur ve bu kümeleri, $\langle i,j \rangle$ ikilisini R_1, R_2 ve R_3 kümelerinden birine atamak için kullan.
5. R_3 kümesindeki her bir $\langle i,j \rangle$ ikilisi için $F(i,j)$ kümesini oluştur ve $\langle i,j \rangle$ ikilisini $(p,q) \in F(i,j)$ koşulunu sağlayan her bir Q_{pq} kümesine ekle.
6. *Küme kapsama* problemi için optimal çözümü bul, diğer bir deyişle, en az sayıda Q_{pq} kümesinin birleşimi ile R_3 kümesini bul.

Algoritmanın 1-5 adımları arasında yapılan dönüşümün algoritma karmaşıklığı $O(n^4)$ 'dir. Algoritmanın 1-4 adımlarının işletim zamanı $O(n^3)$ 'de, 5. adımı ise $O(n^4)$ 'de hesaplanabilir. 5. adımın algoritma karmaşıklığı özenli bir gerçekleştirim ile azaltılabilir, fakat 6. adımdaki küme kapsama probleminin çözümü $\Omega(n^4)$ ile alttan sınırlı olduğundan, bu algoritmanın karmaşıklığı $\Omega(n^4)$ ile alttan sınırlıdır.

DAA'daki öncelik ilişkileri, küme kapsama probleminin polinomsal zamanda çözülmesini sağlayacak özel durumları içeriyor ise, *en az yapay kenar problemi*, polinomsal zamanda çözümlenebilir. Şekil 2.6(a)'da verilen DAA, *en az yapay kenar probleminin* polinomsal zamanda çözüleceği türde öncelik ilişkileri içerir. Bu nedenle, bu ağ için bölüm 2.3.3'de verilen algoritmanın 6. adımı izlenebilir.

Michael'in algoritmasındaki 6. adımın polinomsal zamanda aşılamayacağını anlamak için, diğer bir DAA ile ilgilenelim(Bkz. Şekil 2.8(a)). Çizelge 2.4'de, bu DAA'daki her bir düğüm için $S(v)$, $P^*(v)$, $P^*(s_v)$ ve $P^*(t_v)$ kümeleri verilmiştir; Çizelge 2.5'de ise KAA'da yer alacak olan her bir k düğümü için $P^*(k)$ kümesi ile bu düğümde başlayan ve biten aktivitelerin uç noktaları verilmiştir. Çizelge 2.4 ve 2.5'den yararlanılarak oluşturulan kenar aktif çatı Şekil 2.8(b)'de verilmiştir.



Şekil 2.8 DAA örneği(a) ve bu ağa karşılık gelen kenar aktif çatı(b)

Çizelge 2.4 Şekil 2.8(a)'da yer alan DAA'daki her bir v düğümü için $S(v)$, $P^*(v)$, $P^*(s_v)$ ve $P^*(t_v)$ kümeleri

V	$S(v)$	$P^*(v)$	$P^*(s_v)$	$P^*(t_v)$
1	5,6,7	-	-	1,2
2	5,6,7,8	-	-	2
3	5,6,8	-	-	2,3
4	5,7,8	-	-	2,4
5	-	1,2,3,4	1,2,3,4	1,2,3,4,5,6,7,8
6	-	1,2,3	1,2,3	1,2,3,4,5,6,7,8
7	-	1,2,4	1,2,4	1,2,3,4,5,6,7,8
8	-	2,3,4	2,3,4	1,2,3,4,5,6,7,8

Çizelge 2.5 Şekil 2.8(b)'da yer alan kenar aktif çatıdaki her bir k düğümü için $P^*(k)$ kümesi ve k düğümüne ilişkin aktivitelerin uç noktaları

k	$P^*(k)$	uç noktalar
s	-	s_1, s_2, s_3, s_4
b_1	1,2,3,4	s_5
b_2	1,2,3	s_6
b_3	1,2,4	s_7
b_4	2,3,4	s_8
a_1	1,2	t_1
a_2	2	t_2
a_3	2,3	t_3
a_4	2,4	t_4
f	1,2,3,4,5,6,7,8	t_5, t_6, t_7, t_8

Şekil 2.8(b)'de verilen kenar aktif çatıda yer almayan ilişkiler ve bu ilişkilerin sağlanması için eklenmesi gereken $\langle i, j \rangle$ ikililerini içeren R kümesi aşağıda verilmiştir :

$$\text{İlişkiler} = \{ \langle 1, 5 \rangle, \langle 1, 6 \rangle, \langle 1, 7 \rangle, \langle 2, 5 \rangle, \langle 2, 6 \rangle, \langle 2, 7 \rangle, \langle 2, 8 \rangle, \langle 3, 5 \rangle, \langle 3, 6 \rangle, \langle 3, 8 \rangle, \\ \langle 4, 5 \rangle, \langle 4, 7 \rangle, \langle 4, 8 \rangle \}$$

$$R = \{ \langle a_1, b_1 \rangle, \langle a_1, b_2 \rangle, \langle a_1, b_3 \rangle, \langle a_2, b_1 \rangle, \langle a_2, b_2 \rangle, \langle a_2, b_3 \rangle, \langle a_2, b_4 \rangle, \langle a_3, b_1 \rangle, \langle a_3, b_2 \rangle, \\ \langle a_3, b_4 \rangle, \langle a_4, b_1 \rangle, \langle a_4, b_3 \rangle, \langle a_4, b_4 \rangle \}$$

Çizelge 2.6'da, R kümesindeki her bir (i, j) ikilisi için oluşturulan $X(i, j)$ ve $Y(i, j)$ kümeleri verilmiştir.

Çizelge 2.6 R kümesindeki her bir (i, j) ikilisi için $X(i, j)$, $Y(i, j)$, $X(i, j) \cap Y(i, j)$ ve $X(i, j) \cup Y(i, j)$ kümeleri

R	$X(i, j)$	$Y(i, j)$	$X(i, j) \cap Y(i, j)$	$X(i, j) \cup Y(i, j)$
$\langle a_1, b_1 \rangle$	$\emptyset$	$\{b_2, b_3\}$	$\emptyset$	$\{b_2, b_3\}$
$\langle a_1, b_2 \rangle$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$\langle a_1, b_3 \rangle$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$\langle a_2, b_1 \rangle$	$\{a_1, a_3, a_4\}$	$\{b_2, b_3, b_4\}$	$\emptyset$	$\{a_1, a_3, a_4, b_2, b_3, b_4\}$
$\langle a_2, b_2 \rangle$	$\{a_1, a_3\}$	$\emptyset$	$\emptyset$	$\{a_1, a_3\}$
$\langle a_2, b_3 \rangle$	$\{a_1, a_4\}$	$\emptyset$	$\emptyset$	$\{a_1, a_4\}$
$\langle a_2, b_4 \rangle$	$\{a_3, a_4\}$	$\emptyset$	$\emptyset$	$\{a_3, a_4\}$
$\langle a_3, b_1 \rangle$	$\emptyset$	$\{b_2, b_4\}$	$\emptyset$	$\{b_2, b_4\}$
$\langle a_3, b_2 \rangle$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$\langle a_3, b_4 \rangle$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$\langle a_4, b_1 \rangle$	$\emptyset$	$\{b_3, b_4\}$	$\emptyset$	$\{b_3, b_4\}$
$\langle a_4, b_3 \rangle$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$\langle a_4, b_4 \rangle$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

Çizelge 2.6'da verilen kümelere göre elde edilen R_1 , R_2 ve R_3 kümeleri aşağıda verilmiştir :

$$R_1 = \emptyset$$

$$R_2 = \{\langle a_1, b_2 \rangle, \langle a_1, b_3 \rangle, \langle a_3, b_2 \rangle, \langle a_3, b_4 \rangle, \langle a_4, b_3 \rangle, \langle a_4, b_4 \rangle\}$$

$$R_3 = \{\langle a_1, b_1 \rangle, \langle a_2, b_1 \rangle, \langle a_2, b_2 \rangle, \langle a_2, b_3 \rangle, \langle a_2, b_4 \rangle, \langle a_3, b_1 \rangle, \langle a_4, b_1 \rangle\}$$

R_3 kümesinde yer alan her bir $\langle i, j \rangle$ ikilisi için oluşturulan $F(i, j)$ kümeleri aşağıda verilmiştir.

$$F = \{ F(a_1, b_1), F(a_2, b_1), F(a_2, b_2), F(a_2, b_3), F(a_2, b_4), F(a_3, b_1), F(a_4, b_1) \}$$

$$F(a_1, b_1) = \{ \langle b_2, b_1 \rangle, \langle b_3, b_1 \rangle \}$$

$$F(a_2, b_1) = \{ \langle a_2, a_1 \rangle, \langle a_2, a_3 \rangle, \langle a_2, a_4 \rangle, \langle b_2, b_1 \rangle, \langle b_3, b_1 \rangle, \langle b_4, b_1 \rangle \}$$

$$F(a_2, b_2) = \{ \langle a_2, a_1 \rangle, \langle a_2, a_3 \rangle \}$$

$$F(a_2, b_3) = \{ \langle a_2, a_1 \rangle, \langle a_2, a_4 \rangle \}$$

$$F(a_2, b_4) = \{ \langle a_2, a_3 \rangle, \langle a_2, a_4 \rangle \}$$

$$F(a_3, b_1) = \{ \langle b_2, b_1 \rangle, \langle b_4, b_1 \rangle \}$$

$$F(a_4, b_1) = \{ \langle b_3, b_1 \rangle, \langle b_4, b_1 \rangle \}$$

$F(i, j)$ kümelerinde yer alan (p, q) ikilileri için oluşturulan Q_{pq} kümeleri aşağıda verilmiştir :

$$C = \{ Q(b_2, b_1), Q(b_3, b_1), Q(b_4, b_1), Q(a_2, a_1), Q(a_2, a_3), Q(a_2, a_4) \}$$

$$Q(b_2, b_1) = \{ \langle a_1, b_1 \rangle, \langle a_2, b_1 \rangle, \langle a_3, b_1 \rangle \}$$

$$Q(b_3, b_1) = \{ \langle a_1, b_1 \rangle, \langle a_2, b_1 \rangle, \langle a_4, b_1 \rangle \}$$

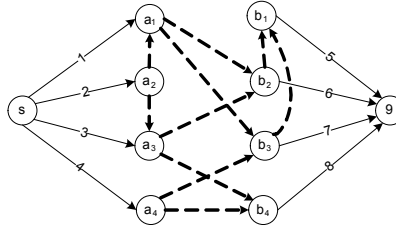
$$Q(b_4, b_1) = \{ \langle a_2, b_1 \rangle, \langle a_3, b_1 \rangle, \langle a_4, b_1 \rangle \}$$

$$Q(a_2, a_1) = \{ \langle a_2, b_1 \rangle, \langle a_2, b_2 \rangle, \langle a_2, b_3 \rangle \}$$

$$Q(a_2, a_3) = \{ \langle a_2, b_1 \rangle, \langle a_2, b_2 \rangle, \langle a_2, b_4 \rangle \}$$

$$Q(a_2, a_4) = \{ \langle a_2, b_1 \rangle, \langle a_2, b_3 \rangle, \langle a_2, b_4 \rangle \}$$

Algoritmanın bu adımında (6. adım), en az sayıda Q_{pq} kümesinin birleşimi ile R_3 kümesi elde edilmeye, *küme kapsama* problemi çözülmeye, çalışılır. *Küme kapsama* probleminin çözümü ile R_3 kümesindeki ilişkileri sağlamak için $\langle b_2, b_1 \rangle$, $\langle b_3, b_1 \rangle$, $\langle a_2, a_1 \rangle$, $\langle a_2, a_3 \rangle$ kenarlarının kenar aktif çatıya eklenmesi gerektiği bulunur. R_2 kümesindeki kenarların da kenar aktif çatıya eklenmesi ile oluşan KAA Şekil 2.9'da verilmiştir.



Şekil 2.9 Şekil 2.8(a)'da yer alan DAA'a karşılık gelen KAA

Şekil 2.8(a)'da verilen DAA, $8(=n=|V|)$ aktivite içermektedir. Söz konusu ağ, n^* adet aktivite içerdiğinde, DAA'ı oluşturmak için izlenmesi gereken yol şöyle verilebilir. Bir projede yer alan $n(=|V|)$ aktivite, $|V_1|=|V_2|$ olacak biçimde iki kümeye ayrılsın. V_1 kümesindeki her bir aktiviteyi/düğümü(u) kuyruk düğüm, V_2 kümesindeki her aktiviteyi/düğümü(v) baş düğüm kabul eden olası tüm *yönlü kenarlar* çizildiğinde, DAA, *tam ikiye ayrılır çizge*(complete bipartite graph) biçimine dönüşür. V_2 kümesinde yer alan aktivitelerin yakın öncellerinin benzer olmamasını sağlayacak biçimde *tam ikiye ayrılır çizgeden* en az sayıda *yönlü kenar* çıkartılır ise, ara çizgede kenar sayısını maksimum yapacak bir DAA elde edilmiş olur⁴. Dolayısıyla, Şekil 2.8(a)'da verilen DAA daha büyük n değerleri için çizilebilecektir. n adet aktivitenin öncelik ilişkilerinin yukarıda anlatıldığı gibi düzenlenmesi durumunda, Michael ve diğ.'nin önerdiği yaklaşımın 6. adımında her bir Q_{pq} kümesi $\frac{n}{2}-1$ eleman içerecektir ve $n(=|C|)$ adet Q_{pq} kümesi ortaya çıkacaktır. R_3 kümesini bulmak için "hangi Q_{pq} kümeleri ele alınmalıdır" sorusunu⁵ yanıtlayan bir algoritma geliştirilse bile "izlenebilir" bir algoritma olmayacaktır. Oysa, tezde verilen yaklaşım, söz konusu DAA'ı $O(n^4)$ 'te Şekil 2.9'da verilen KAA'a dönüştürür.

2.4 Sharma'nın Yaklaşımı

Bu çalışmada, *En Az Yapay Kenar Problemine* girmeden KAA'ı kuran sezgisel bir yaklaşım(heuristic approach) verilmiştir. Bu yaklaşımda, proje girdisindeki her bir $i;j$ ikilisi teker teker ele alınır. Eğer ilgili $i;j$ ilişkisi kenar aktif ağda yer almıyor ise, bu ilişkiyi kurmak için ağa üç tane yapay kenar eklenir ve daha sonra eklenen bu yapay kenarların her biri büzülmeye çalışılır. Sharma tarafından önerilen algoritma, [11] no'lu çalışmada ayrıntılı olarak incelenmiştir.

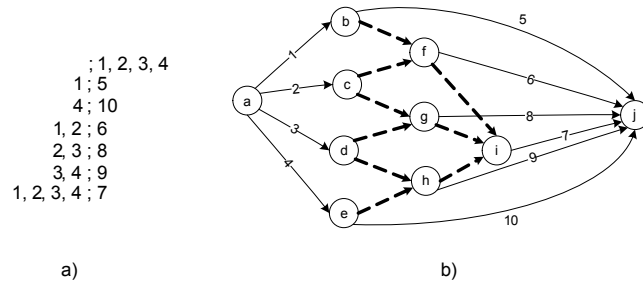
* n değerinin çift sayı olduğunu kabul edelim

⁴ Bu konunun ayrıntıları, [7] numaralı çalışmada bulunabilir.

⁵ Bu sorun, küme kapsama problemidir.

2.4.1 Fazlalık Aktiviteler

Sharma tarafından önerilen algoritma sonucunda fazlalık aktivite içeren ağların oluşabileceği Güngör[11]'ün çalışmasında gösterilmiştir. Şekil 2.10'da, [11] no'lu çalışmadan alınan bir proje girdisi ve bu proje girdisi için Sharma'nın algoritmasının ürettiği KAA verilmiştir. Şekil 2.10(a)'daki proje girdisinde, 7 numaralı aktivitenin öncellerinin 1,2,3 ve 4 numaralı aktiviteler olduğu görülmektedir. Bu öncelik ilişkisinin kurulması için Şekil 2.10(b)'de yer alan ağdaki $\langle f,i \rangle$ ve $\langle h,i \rangle$ yapay kenarları yeterlidir, $\langle g,i \rangle$ yapay kenarı, 7 numaralı aktivitenin öncel kümesine bir katkıda bulunmaz. Dolayısıyla $\langle g,i \rangle$ yapay kenarı fazlalıktır, bu kenarın ağdan çıkartılması öncelik ilişkilerini bozmaz.



Şekil 2.10 Bir proje girdisi(a) ve bu proje girdisi için Sharma'nın algoritmasının ürettiği KAA(b)

2.4.2 Algoritma Analizi

n , düğüm aktif ağdaki düğüm sayısı(aktivite sayısı); $m(m=n/c, c$ sabit bir sayı) ise, bir aktiviteye ait olabilecek en fazla öncel aktivite sayısı olsun. [18] no'lu çalışmada, algoritma karmaşıklığının $O(n^2 \log n)$ ile sınırlı olduğu belirtilmiştir. [11] no'lu çalışmada ise algoritma karmaşıklığının $O(n^2 \log n)$ değil, $O(n^2 m^2)$ olduğu ispatlanmıştır. Dolayısıyla, Sharma'nın yaklaşımında önerilen algoritma $O(n^2 m^2)$ ile sınırlıdır.

3. KENAR AKTİF AĞ OLUŞTURMAK İÇİN ÖNERİLEN YAKLAŞIM

Bu tez çalışmasında, olabildiğince az sayıda yapay aktivite içeren KAA'nın kurulması için bir yaklaşım önerilmiştir. Öncelikle DAA'dan KAA'a hızlı bir dönüşüm için Ünlü'nün önerdiği ön işleme algoritması kullanılmıştır. Ünlü'nün algoritmasının uygulanması sonucu elde edilen KAA üzerinde, Güngör'ün önerdiği ve bu tez çalışmasında yeniden gözden geçirilerek geliştirilen indirgeme kuralları uygulanarak ağdaki yapay aktivite sayısı azaltılmaya çalışılmıştır. En kötü durumda polinomsal süreden daha fazla zaman gereksinimi duymayan ve küme kapsama problemi çözümünü de içermeyen indirgeme kuralları dört ayrı sınıf altında incelenmiştir. Önerilen yaklaşımın etkili gerçekleştirimi için indirgeme kurallarının işleniş sırası ve kullanılacak veri yapıları da vurgulanmıştır.

3.1 İndirgeme Kuralları

Bu çalışmada geliştirilen indirgeme kuralları, dayalı oldukları mantığa göre dört sınıfa ayrılır:

1. Yapay Kenar Büzme Kuralları
2. Düğüm Örtüşme Kuralları
3. Yapay Kenar Ekleme Kuralları
4. Düğüm Ekleme Kuralları

İndirgeme kurallarında ele alınan kenar aktif ağ $G=(V,E,D)$ ile ifade edilsin. V , düğümler kümesini; E , gerçel kenarlar kümesini; D , yapay kenarlar kümesini temsil etsin. İndirgeme kurallarında kullanılan semboller aşağıda verilmiştir :

$P_x = \{p_i \mid \langle p_i, x \rangle \in D\}$: Baş düğümü x olan yapay kenarların kuyruk düğümleri.

$\bar{P}_x = \{p_i \mid \langle p_i, x \rangle \in E\}$: Baş düğümü x olan gerçel kenarların kuyruk düğümleri.

$$P_x = P_x \cup \bar{P}_x$$

$S_x = \{s_i \mid \langle x, s_i \rangle \in D\}$: Kuyruk düğümü x olan yapay kenarların baş düğümleri.

$\bar{S}_x = \{s_i \mid \langle x, s_i \rangle \in E\}$: Kuyruk düğümü x olan gerçel kenarların baş düğümleri.

$$S_x = S_x \cup \bar{S}_x$$

3.1.1 Yapay Kenar Büzme Kuralları

G çizgesinde bir yapay kenar $\langle x, y \rangle \in D$ ile ilgilenelim. Bu sınıfa giren kurallar, x ve y düğümlerinden birinin, diğerinin üzerine gelinceye değin yapay kenarın bir anlamda büzülmesi temel mantığına dayanır. Kurala bağlı olarak, büzme işleminden sonra paralel kenar oluşmaması için gerekli denetim yapılır. Büzme sonrası, düğümlerden biri düğümler kümesinden, $\langle x, y \rangle$ yapay kenarı da yapay kenarlar kümesinden çıkartılır. Önceliklerin bozulmaması için kurala bağlı olarak gerekli kuyruk ve baş düzenlemeleri yapılır. Yapay kenar büzme kurallarında, Bölüm 3.1’de verilen kümelere ek olarak aşağıdaki kümeler de kullanılır.

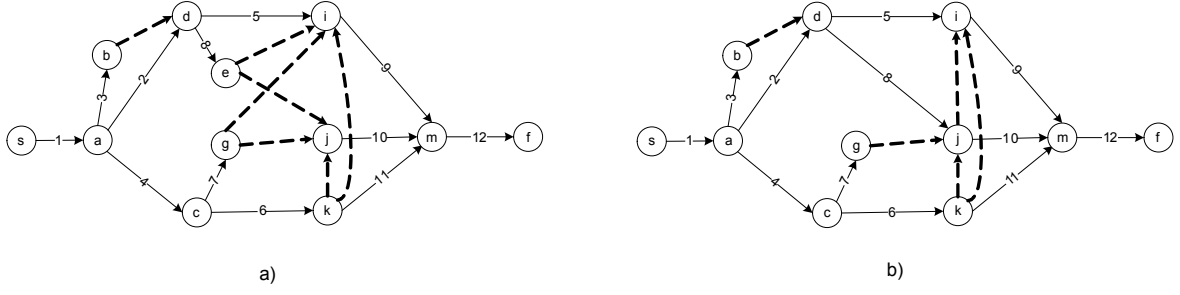
$K = \{k_i \mid \langle x, k_i \rangle \in D\} - \{y\} : y$ dışında, kuyruk düğümü x olan yapay kenarların baş düğümleri.

$L = \{l_i \mid \langle l_i, y \rangle \in D\} - \{x\} : x$ dışında, baş düğümü y olan yapay kenarların kuyruk düğümleri.

3.1.1.1 1. Yapay Kenar Büzme Kuralı

G çizgesinde kuyruk düğümü x , baş düğümü y olan bir yapay kenar $\langle x, y \rangle \in D$ ile ilgilenelim. x düğümünü kuyruk, y düğümünü baş düğüm kabul eden hiçbir gerçel kenar olmasın ($\bar{S}_x = \bar{P}_y = \emptyset$). x düğümünü baş ya da/ve y düğümünü kuyruk kabul eden kenarların olup olmaması bu kuralın uygulanmasında etkili değildir. K ve L kümeleri boş değil ve $L \times K$ kartezyan kümesinin her bir elemanı (l, k) , G çizgesinde yapay ya da gerçel kenar olarak yer alıyor ise 1. büzme kuralı uygulanır.

Şekil 3.1’de, 1. büzme kuralının uygulanmasına ilişkin bir örnek verilmiştir. Şekil 3.1(a)’da verilen çizgede $\langle e(=x), j(=y) \rangle$ yapay kenarı ile ilgilenelim. Öncelikle, K ve L kümeleri bulunmalıdır. e no’lu düğümden j no’lu düğüm dışında yalnızca i no’lu düğüme gidilmektedir. Bu durumda $K = \{i\}$ olacaktır. j no’lu düğüme e no’lu düğüm dışında g ve k no’lu düğümlerden gelinmektedir, dolayısıyla $L = \{g, k\}$ bulunur. $\langle g, i \rangle$ ve $\langle k, i \rangle$ kenarlarının(yapay ya da gerçel) G çizgesinde bulunması nedeni ile $\langle e, j \rangle$ yapay kenarı üzerinde 1. büzme kuralı uygulanabilir. Şekil 3.1(a)’da verilen çizgede $\langle e, j \rangle$ yapay kenarının büzülmesi sonucu oluşan çizge, Şekil 3.1(b)’de verilmiştir.



Şekil 3.1 [21] no'lu çalışmadan alınan bir çizge örneği(a) ve bu çizgedeki $\langle e,j \rangle$ yapay kenarının 1. büzme kuralına göre büzülmesi sonucu oluşan çizge(b)

1. yapay kenar büzme kuralına ilişkin yordam aşağıda verilmiştir :

```

Yapay_Kenar_Büzme_Kuralı_1( $x,y$ )
{
  if ( ( $\langle x,y \rangle \in D$ )  $\wedge$  ( $\bar{S}_x = \bar{P}_y = \emptyset$ )  $\wedge$  ( $K \neq \emptyset \wedge L \neq \emptyset$ ))
  {
    if ( ( $\forall_l \forall_k [ \langle l,k \rangle \in D \vee \langle l,k \rangle \in E : l \in L, k \in K ]$ ) ) // kartezyen çarpımda yer alan
    {
      for (each  $l \in L$ ) // her bir  $(l,k)$  ikilisi,  $G$  çizgesinde
      {
        for (each  $k \in K$ ) // bir kenar(yapay ya da gerçel) olarak var mı?
        {
          if (  $\langle l,k \rangle \in D$  ) //  $(l,k)$  ikilisi,  $G$  çizgesinde yapay kenar
          {
             $D = D - \{ \langle l,k \rangle \}$  // ise çizgeden çıkart
          }
          // ilgilenilen yapay kenarı  $\langle x,y \rangle$  çizgeden çıkart
          //  $K$  kümesindeki düğümleri baş düğüm kabul
          // eden ve kuyruğu  $x$  olan kenarların kuyruk
          // düğümlerini  $y$  düğümü yap
          //  $x$  düğümüne gelen yapay kenarları
          //  $y$  düğümüne yönlendir.
           $D = D - \{ \langle d,x \rangle \}$ 
           $D = D \cup \{ \langle d,y \rangle \}$ 
        }
      }
      for (each  $r \in \bar{P}_x$ ) //  $x$  düğümüne gelen gerçel kenarları
      {
         $E = E - \{ \langle r,x \rangle \}$ 
         $E = E \cup \{ \langle r,y \rangle \}$ 
        //  $y$  düğümüne yönlendir.
      }
       $V = V - \{ x \}$ 
    }
  }
}

```

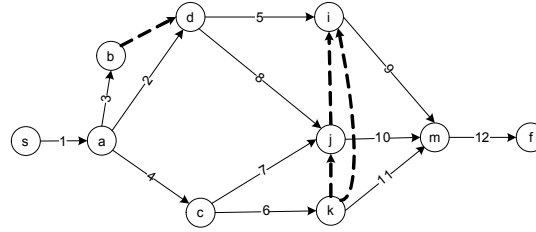
“Yapay_Kenar_Büzme_Kuralı_1” yordamında yer alan ikinci “if” komutundaki (l,k) ikililerinin denetiminin karmaşıklığı $O(n^2)$ olarak verilebilir. Yordamda bulunan iç içe iki “for” döngüsünün karmaşıklığının denetimden bağımsız ve $O(n^2)$ olması nedeni ile, 1. büzme kuralının algoritma karmaşıklığı, $O(n^2)$ olacaktır.

3.1.1.2 2. Yapay Kenar Büzme Kuralı

G çizgesinde kuyruk düğümü x , baş düğümü y olan bir yapay kenar($\langle x,y \rangle \in D$) ile ilgilenelim. x düğümünü kuyruk kabul eden hiçbir gerçel kenar olmasın($\bar{S}_x = \emptyset$) ve x düğümünü kuyruk kabul eden sadece bir yapay kenar olsun($K = \emptyset$). y düğümünü baş ya da/ve kuyruk kabul eden kenarların olup olmaması bu kuralın

uygulanmasında etkili değildir. $\bar{P}_x \cap \bar{P}_y = \emptyset$ koşulu sağlanıyor ise, 2. büzme kuralı uygulanır. $\bar{P}_x \cap \bar{P}_y = \emptyset$ koşulu ile büzme sonrası paralel kenar oluşmaması sağlanır.

Şekil 3.2’de, 2. büzme kuralının uygulanmasına ilişkin bir örnek verilmiştir. Şekil 3.1(b)’de verilen çizgede $\langle g(=x), j(=y) \rangle$ yapay kenarı ile ilgilenelim. $\langle g,j \rangle$ yapay kenarı, g düğümünü kuyruk kabul eden tek kenardır. Baş düğümü g olan gerçel kenarların kuyruk düğümleri ($\bar{P}_g = \{c\}$) ile baş düğümü j olan gerçel kenarların kuyruk düğümleri ($\bar{P}_j = \{d\}$) ortak herhangi bir düğüm içermez ($\bar{P}_g \cap \bar{P}_j = \emptyset$). Bu durumda $\langle g,j \rangle$ yapay kenarı üzerinde 2. büzme kuralı uygulanabilir. Şekil 3.1(b)’de verilen çizgedeki $\langle g,j \rangle$ yapay kenarının büzülmesi sonucu oluşan çizge Şekil 3.2’de verilmiştir.



Şekil 3.2 Şekil 3.1(b)’de yer alan çizgedeki $\langle g,j \rangle$ yapay kenarının 2. büzme kuralına göre büzülmesi sonucu oluşan çizge

2. yapay kenar büzme kuralına ilişkin yordam aşağıda verilmiştir :

```

Yapay_Kenar_Büzme_Kuralı_2( $x,y$ )
{
  if ( $((\langle x,y \rangle \in D) \wedge (\bar{S}_x = \emptyset \wedge \bar{S}_y = \{y\})) \wedge (\bar{P}_x \cap \bar{P}_y = \emptyset)$ )
  {
     $D = D - \{\langle x,y \rangle\}$  // ilgililenen yapay kenarı  $\langle x,y \rangle$  çizgeden çıkart
    for (each  $d \in \bar{P}_x$ ) // x düğümüne gelen yapay kenarları
       $D = D - \{\langle d,x \rangle\}$  // y düğümüne yönlendir
       $D = D \cup \{\langle d,y \rangle\}$ 
    for (each  $r \in \bar{P}_y$ ) // x düğümüne gelen gerçel kenarları
       $E = E - \{\langle r,x \rangle\}$  // y düğümüne yönlendir
       $E = E \cup \{\langle r,y \rangle\}$ 
     $V = V - \{x\}$ 
  }
}

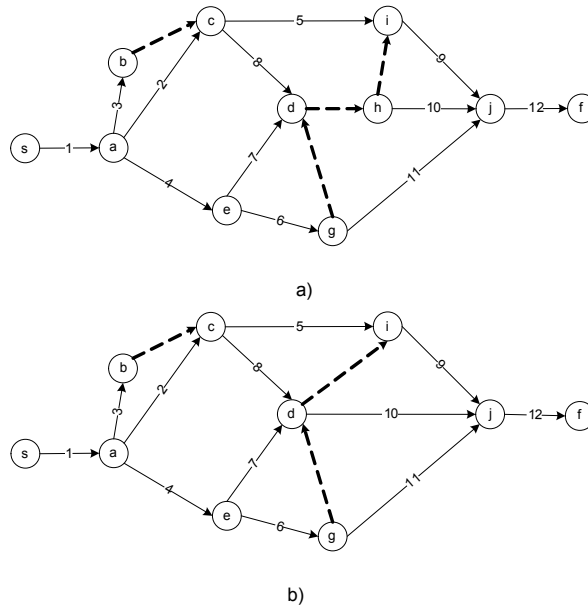
```

“Yapay_Kenar_Büzme_Kuralı_2” yordamında $\bar{P}_x \cap \bar{P}_y = \emptyset$ denetimi $O(n)$ ’de gerçekleştirilir. Yordamdaki döngülerin gereksinim duyacağı işletim süresi de $O(n)$ ile sınırlıdır ve denetimden bağımsızdır, dolayısıyla 2. büzme kuralı için karmaşıklık $O(n)$ olarak verilebilir.

3.1.1.3 3. Yapay Kenar Büzme Kuralı

G çizgesinde kuyruk düğümü x , baş düğümü y olan bir yapay kenar ($\langle x, y \rangle \in D$) ile ilgilenelim. y düğümünü baş kabul eden hiç gerçel kenar olmasın ($\bar{P}_y = \emptyset$) ve y düğümünü baş kabul eden sadece bir yapay kenar olsun ($L = \emptyset$). x düğümünü baş ya da/ve kuyruk kabul eden kenarların olup olmaması bu kuralın uygulanmasında etkili değildir. $\bar{S}_x \cap \bar{S}_y = \emptyset$ koşulu da sağlanıyor ise 3. büzme kuralı uygulanır. $\bar{S}_x \cap \bar{S}_y = \emptyset$ koşulu ile büzme sonrası paralel kenar oluşmaması sağlanır.

Şekil 3.3'de, 3. büzme kuralının uygulanmasına ilişkin bir örnek verilmiştir. Şekil 3.3(a)'da verilen çizgede $\langle d(=x), h(=y) \rangle$ yapay kenarı ile ilgilenelim. $\langle d, h \rangle$ yapay kenarı, h düğümünü baş kabul eden tek kenardır. Kuyruk düğümü d olan gerçel kenarların baş düğümleri ($\bar{S}_d = \emptyset$) ile kuyruk düğümü h olan gerçel kenarların baş düğümleri ($\bar{S}_h = \{j\}$) ortak herhangi bir düğüm içermez ($\bar{S}_d \cap \bar{S}_h = \emptyset$). Bu durumda $\langle d, h \rangle$ yapay kenarı üzerinde 3. büzme kuralı uygulanabilir. Şekil 3.3(a)'da verilen çizgedeki $\langle d, h \rangle$ yapay kenarının büzülmesi sonucu oluşan çizge Şekil 3.3(b)'de verilmiştir.



Şekil 3.3 [11] no'lu çalışmadan alınan bir çizge örneği(a) ve bu çizgedeki $\langle d, h \rangle$ yapay kenarının 3. büzme kuralına göre büzülmesi sonucu oluşan çizge(b)

3. yapay kenar büzme kuralına ilişkin yordam aşağıda verilmiştir :

```

Yapay_Kenar_Büzme_Kuralı_3( $x, y$ )
{
  if (( $\langle x, y \rangle \in D$ ) )  $\wedge$  ( $\bar{P}_y = \emptyset \wedge \bar{P}_x = \{x\}$ )  $\wedge$  ( $\bar{S}_x \cap \bar{S}_y = \emptyset$ )
  {
     $D = D - \{\langle x, y \rangle\}$  // ilgilenilen yapay kenarı çizgeden çıkart
    for (each  $z \in \bar{P}_y$ ) // y düğümünden çıkan yapay kenarların
    {
       $D = D - \{\langle y, z \rangle\}$  // kuyruk düğümlerini x düğümü yap.
       $D = D \cup \{\langle x, z \rangle\}$ 
    }
    for (each  $s \in \bar{S}_y$ ) // y düğümünden çıkan gerçel kenarların
    {
       $E = E - \{\langle y, s \rangle\}$  // kuyruk düğümlerini x düğümü yap
       $E = E \cup \{\langle x, s \rangle\}$ 
    }
     $V = V - \{y\}$ 
  }
}

```

3. yapay kenar büzme kuralı, 2. yapay kenar büzme kuralının simetriği olması nedeni ile “Yapay_Kenar_Büzme_Kuralı_3” yordamının algoritma karmaşıklığı $O(n)$ ile üstten sınırlıdır.

3.1.2 Düğüm Örtüşme Kuralları

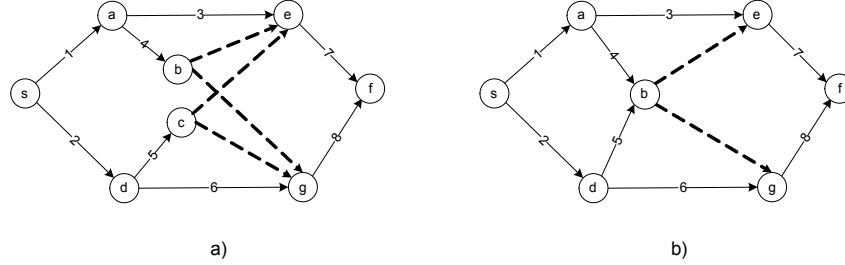
G çizgesindeki x ve y düğümleri ile ilgilenelim. Bu sınıfa giren kurallar, bir düğümün diğer düğüm üzerine örtüştürülmesi temel mantığına dayanır. Bu sınıfa giren kurallar ile çizgedeki düğüm ve yapay kenar sayısı azaltılır.

3.1.2.1 1. Düğüm Örtüşme Kuralı

G çizgesindeki x ve y düğümleri ile ilgilenelim. x ve y düğümlerini kuyruk kabul eden hiçbir gerçel kenar olmasın ($\bar{S}_x = \bar{S}_y = \emptyset$). x ve y düğümlerini kuyruk kabul eden en az ikişer yapay kenar olsun ve bu kenarların baş düğümleri birbirine eşit olsun ($\bar{P}_x = \bar{P}_y$). Büzme sonrası paralel kenar oluşmaması için yapılan $\bar{P}_x \cap \bar{P}_y = \emptyset$ denetimi de sağlanıyor ise, x ve y düğümleri üzerinde 1. düğüm örtüşme kuralı uygulanır.

Şekil 3.4’te, 1. örtüşme kuralının uygulanmasına ilişkin bir örnek verilmiştir. Şekil 3.4(a)’da verilen çizgedeki $b(=x)$ ve $c(=y)$ düğümleri ile ilgilenelim. İlgili düğümleri kuyruk kabul eden hiçbir gerçel kenar yoktur ve bu düğümleri kuyruk kabul eden yapay kenarların baş düğümleri birbirine eşittir ($\bar{P}_b = \bar{P}_c = \{e, g\}$). Baş düğümü b olan gerçel kenarların kuyruk düğümleri ($\bar{P}_b = \{a\}$) ile baş düğümü c olan gerçel kenarların kuyruk düğümleri ($\bar{P}_c = \{d\}$) ortak herhangi bir düğüm içermediğinden ($\bar{P}_b \cap \bar{P}_c = \emptyset$), b ve c düğümleri üzerinde 1. örtüşme kuralı uygulanır. Bu durumda, b düğümü ve bu

düğümü kuyruk kabul eden yapay kenarlar çizgeden çıkarılır ve b düğümünü baş düğüm kabul eden kenarlar c düğümüne yönlendirilir. Şekil 3.4(a)'da verilen çizgedeki c düğümünün b düğümü üzerine örtüştürülmesi sonucu oluşan çizge Şekil 3.4(b)'de verilmiştir.



Şekil 3.4 [11] no'lu çalışmadan alınan bir çizge örneği(a) ve bu çizgedeki b ve c düğümleri üzerinde 1. örtüşme kuralının uygulanması sonucu oluşan çizge(b)

1. düğüm örtüşme kuralına ilişkin yordam aşağıda verilmiştir :

```

Düğüm_Örtüşme_Kuralı_1(x,y)
{
  if (( $\bar{S}_x = \bar{S}_y = \emptyset$ )  $\wedge$  ( $\bar{S}_x = \bar{S}_y$ )  $\wedge$  ( $\bar{P}_x \cap \bar{P}_y = \emptyset$ ))
    for (each  $z \in \bar{S}_x$ ) // x düğümünden çıkan yapay kenarları
       $D = D - \{<x,z>\}$  // çizgeden çıkart
    for (each  $r \in \bar{P}_x$ ) // x düğümüne gelen gerçel kenarları
       $E = E - \{<r,x>\}$  // y düğümüne yönlendir
       $E = E \cup \{<r,y>\}$ 
    for (each  $d \in \bar{S}_y$ ) // x düğümüne gelen yapay kenarları
       $D = D - \{<d,x>\}$  // y düğümüne yönlendir
       $D = D \cup \{<d,y>\}$ 
     $V = V - \{x\}$ 
}

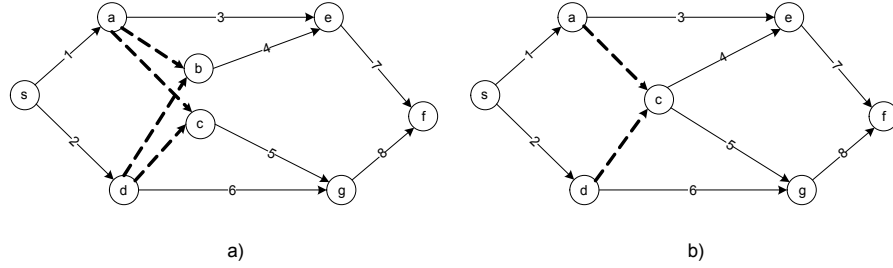
```

Küme eşitliği ya da kesişiminin denetimi $O(n)$ ile sınırlıdır. Döngülerin karmaşıklığının da $O(n)$ olması nedeni ile “Düğüm_Örtüşme_Kuralı_1” yordamının işletim süresi $O(n)$ ile üstten sınırlıdır.

3.1.2.2 2. Düğüm Örtüşme Kuralı

G çizgesindeki x ve y düğümleri ile ilgilenelim. x ve y düğümlerini baş kabul eden hiçbir gerçel kenar olmasın($\bar{P}_x = \bar{P}_y = \emptyset$). x ve y düğümlerini baş kabul eden en az ikişer yapay kenar olsun ve bu kenarların kuyruk düğümleri birbirine eşit olsun($\bar{S}_x = \bar{S}_y$). Büzme sonrası paralel kenar oluşmaması için yapılan $\bar{S}_x \cap \bar{S}_y = \emptyset$ denetimi de sağlanıyor ise, x ve y düğümleri üzerinde 2. örtüşme kuralı uygulanır.

Şekil 3.5'te, 2. örtüşme kuralının uygulanmasına ilişkin bir örnek verilmiştir. Şekil 3.5(a)'da verilen çizgedeki $b(=x)$ ve $c(=y)$ düğümleri ile ilgilenelim. İlgili düğümleri baş kabul eden hiçbir gerçel kenar yoktur ve bu düğümleri baş kabul eden yapay kenarların kuyruk düğümleri birbirine eşittir($\bar{P}_b = \bar{P}_c = \{a, d\}$). Kuyruk düğümü b olan gerçel kenarların baş düğümleri($\bar{S}_b = \{e\}$) ile kuyruk düğümü c olan gerçel kenarların baş düğümleri($\bar{S}_c = \{g\}$) ortak herhangi bir düğüm içermediğinden($\bar{S}_b \cap \bar{S}_c = \emptyset$), b ve c düğümleri üzerinde 2. örtüşme kuralı uygulanır. Bu durumda, b düğümü ve bu düğümü baş kabul eden yapay kenarlar çizgeden çıkarılır ve b düğümünü kuyruk düğüm kabul eden kenarlar c düğümüne yönlendirilir. Şekil 3.5(a)'da verilen çizgedeki c düğümünün b düğümü üzerine örtüştürülmesi sonucu oluşan çizge Şekil 3.5(b)'de verilmiştir.



Şekil 3.5 [11] no'lu çalışmadan alınan bir çizge örneği(a) ve bu çizgedeki b ve c düğümleri üzerinde 2. örtüşme kuralının uygulanması sonucu oluşan çizge(b)

2. düğüm örtüşme kuralına ilişkin yordam aşağıda verilmiştir :

```

Düğüm_Örtüşme_Kuralı_2( $x, y$ )
{
  if ( $((\bar{P}_x = \bar{P}_y = \emptyset) \wedge (\bar{P}_x = \bar{P}_y) \wedge (\bar{S}_x \cap \bar{S}_y = \emptyset))$ )
  {
    for (each  $z \in \bar{P}_x$ ) //  $x$  düğümüne gelen yapay kenarları
       $D = D - \{<z, x>\}$  // çizgeden çıkart
    for (each  $d \in \bar{P}_y$ ) //  $x$  düğümünden çıkan yapay kenarların
       $D = D - \{<x, d>\}$  // kuyruk düğümlerini  $y$  düğümü yap.
       $D = D \cup \{<y, d>\}$ 
    for (each  $r \in \bar{S}_x$ ) //  $x$  düğümünden çıkan gerçel kenarların
       $E = E - \{<x, r>\}$  // kuyruk düğümlerini  $y$  düğümü yap
       $E = E \cup \{<y, r>\}$ 
     $V = V - \{x\}$ 
  }
}

```

2. düğüm örtüşme kuralı, 1. düğüm örtüşme kuralının simetriği olması nedeni ile “Düğüm_Örtüşme_Kuralı_2” yordamının algoritma karmaşıklığı $O(n)$ ile üstten sınırlıdır.

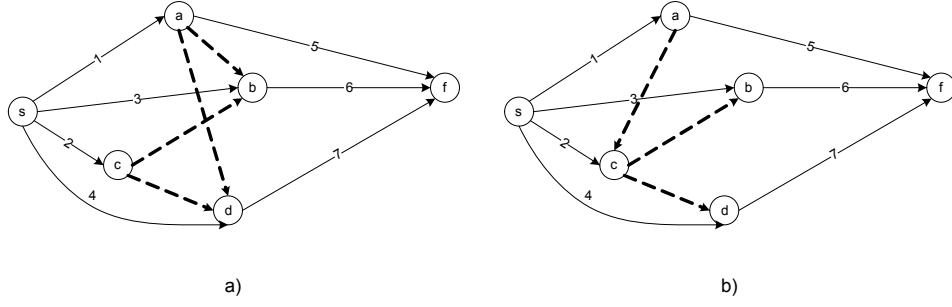
3.1.3 Yapay Kenar Ekleme Kuralları

G çizgesindeki x ve y düğümleri ile ilgilenelim. Bu iki düğüm arasında yapay ya da gerçel kenar olmasın. Bu sınıfa giren kurallar, x ve/ya da y düğümleri ile ilişkili en az iki yapay kenarın silinip, $\langle x,y \rangle$ yapay kenarının eklenerek öncelik ilişkilerinin korunması mantığına dayanır. En az iki yapay kenar silinmesinin amacı; kenar aktif bir çizgede yapay kenar ekleme kuralı uygulandıktan sonra çizgede bulunan yapay kenar sayısının azalmasını sağlamaktır.

3.1.3.1 1. Yapay Kenar Ekleme Kuralı

G çizgesinde x ve y düğümleri ($x,y \in V$) ile ilgilenelim. y düğümünü kuyruk kabul eden hiçbir gerçel kenar olmasın ($\bar{S}_y = \emptyset$) ve y düğümünü kuyruk kabul eden en az iki yapay kenar olsun ($|S_y| > 1$). x düğümünü kuyruk kabul eden gerçel kenarların olup olmaması bu kuralın uygulanmasında etkili değildir. x ve y düğümleri arasında yapay ya da gerçel kenar yok ve y düğümünü kuyruk kabul eden yapay kenarların baş düğümleri, x düğümünü kuyruk kabul eden yapay kenarların baş düğümlerinin bir alt kümesi ($S_y \subseteq S_x$) ise 1. yapay kenar ekleme kuralı uygulanır.

Şekil 3.6'da, 1. yapay kenar ekleme kuralının uygulanmasına ilişkin bir örnek verilmiştir. Şekil 3.6(a)'da verilen çizgedeki $a(=x)$ ve $c(=y)$ düğümleri ile ilgilenelim. c düğümünü kuyruk kabul eden hiçbir gerçel kenar bulunmamaktadır. Kuyruk düğümü c olan yapay kenarların baş düğümleri ($S_c = \{b,d\}$), kuyruk düğümü a olan yapay kenarların baş düğümlerinin ($S_a = \{b,d\}$) alt kümesidir ($S_c \subseteq S_a$). a ve c düğümleri arasında yapay ya da gerçel kenar olmadığından, bu düğümler üzerinde 1. yapay kenar ekleme kuralı uygulanır. Bu durumda, a düğümünü kuyruk kabul eden yapay kenarlardan baş düğümleri, c düğümünü kuyruk kabul eden yapay kenarların baş düğümlerine eşit olanlar çizgeden çıkarılır ($\{\langle a,z \rangle \mid z \in S_c\}$) ve $\langle a,c \rangle$ yapay kenarı çizgeye eklenir. Şekil 3.6(a)'da verilen çizgeye $\langle a,c \rangle$ yapay kenarının eklenmesi sonucu oluşan çizge Şekil 3.6(b)'de verilmiştir.



Şekil 3.6 [20] no'lu çalışmadan alınan bir çizge örneği(a) ve bu çizgedeki *a* ve *c* düğümleri arasında 1. yapay kenar ekleme kuralının uygulanması sonucu oluşan çizge(b)

1. yapay kenar ekleme kuralına ilişkin yordam aşağıda verilmiştir :

```

Yapay_Kenar_Ekleme_Kuralı_1(x,y)
{
  if (( $\bar{S}_y = \emptyset \wedge |\bar{S}_y| > 1$ )  $\wedge$  ( $\langle x,y \rangle \notin D \wedge \langle y,x \rangle \notin D \wedge \langle x,y \rangle \notin E$ )  $\wedge$  ( $\bar{S}_y \subseteq \bar{S}_x$ ))
    for (each  $z \in \bar{S}_y$ )           // x düğümünü kuyruk kabul eden
       $D = D - \{ \langle x,z \rangle \}$        // ve baş düğümünü  $\bar{S}_y$  den alan yapay kenarları çizgeden çıkart
       $D = D \cup \{ \langle x,y \rangle \}$     // kuyruk düğümü x, baş düğümü y olan yeni bir yapay kenar ekle
}

```

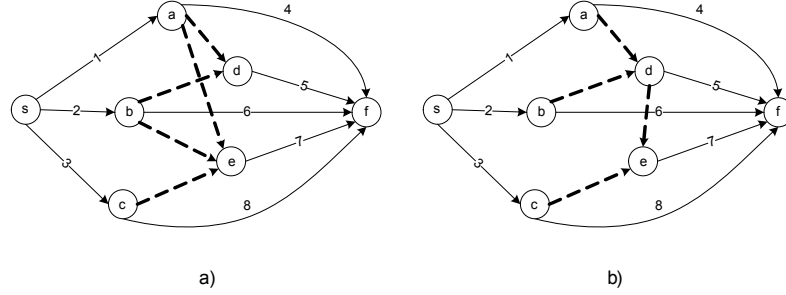
$\bar{S}_y \subseteq \bar{S}_x$ denetimi $O(n)$ 'de gerçekleştirilir. Yordamdaki döngünün de gereksinim duyacağı zaman doğrusaldır, dolayısıyla “Yapay_Kenar_Ekleme_Kuralı_1” yordamı $O(n)$ ile üstten sınırlıdır.

3.1.3.2 2. Yapay Kenar Ekleme Kuralı

G çizgesinde x ve y düğümleri ($x, y \in V$) ile ilgilenelim. x düğümünü baş kabul eden hiçbir gerçel kenar olmasın ($\bar{P}_x = \emptyset$) ve x düğümünü baş kabul eden en az iki yapay kenar olsun ($|\bar{S}_x| > 1$). y düğümünü baş kabul eden gerçel kenarların olup olmaması bu kuralın uygulanmasında etkili değildir. x ve y düğümleri arasında yapay ya da gerçel kenar yok ve x düğümünü baş kabul eden yapay kenarların kuyruk düğümleri, y düğümünü baş kabul eden yapay kenarların kuyruk düğümlerinin bir alt kümesi ($\bar{P}_x \subseteq \bar{P}_y$) ise 2. yapay kenar ekleme kuralı uygulanır.

Şekil 3.7’de, 2. yapay kenar ekleme kuralının uygulanmasına ilişkin bir örnek verilmiştir. Şekil 3.7(a)’da verilen çizgedeki $d(=x)$ ve $e(=y)$ düğümleri ile ilgilenelim. d düğümünü baş kabul eden hiçbir gerçel kenar bulunmamaktadır. Baş düğümü e olan yapay kenarların kuyruk düğümleri ($\bar{P}_e = \{a, b, c\}$), baş düğümü e olan yapay kenarların kuyruk düğümlerinin ($\bar{P}_d = \{a, b\}$) alt kümesidir ($\bar{P}_d \subseteq \bar{P}_e$). d ve e düğümleri

arasında yapay ya da gerçel aktivite olmadığından, bu düğümler üzerinde 2. yapay kenar ekleme kuralı uygulanır. Bu durumda, e düğümünü baş kabul eden yapay kenarlardan kuyruk düğümleri, d düğümünü baş kabul eden yapay kenarların kuyruk düğümlerine eşit olanlar çizgeden çıkarılır($\{<z, e> \mid z \in P_d\}$) ve $<d, e>$ yapay kenarı çizgeye eklenir. Şekil 3.7(a)'da verilen çizgeye $<d, e>$ yapay kenarının eklenmesi sonucu oluşan çizge Şekil 3.7(b)'de verilmiştir.



Şekil 3.7 [4] no'lu çalışmadan alınan bir çizge örneği(a) ve bu çizgedeki d ve e düğümleri arasında 2. yapay kenar ekleme kuralının uygulanması sonucu oluşan çizge(b)

2. yapay kenar ekleme kuralına ilişkin yordam aşağıda verilmiştir :

```

Yapay_Kenar_Ekleme_Kuralı_2(x,y)
{
  if (( $\bar{P}_x = \emptyset \wedge |P_x| > 1$ )  $\wedge$  ( $<x,y> \notin D \wedge <y,x> \notin D \wedge <x,y> \notin E$ )  $\wedge$  ( $P_x \subseteq P_y$ ))
    for (each  $z \in P_x$ ) // y düğümünü baş kabul eden ve kuyruk düğümünü  $P_x$  den
       $D = D - \{<z,y>\}$  // alan yapay kenarları çizgeden çıkart
       $D = D \cup \{<x,y>\}$  // kuyruk düğümü x, baş düğümü y olan yeni bir yapay kenar ekle
}

```

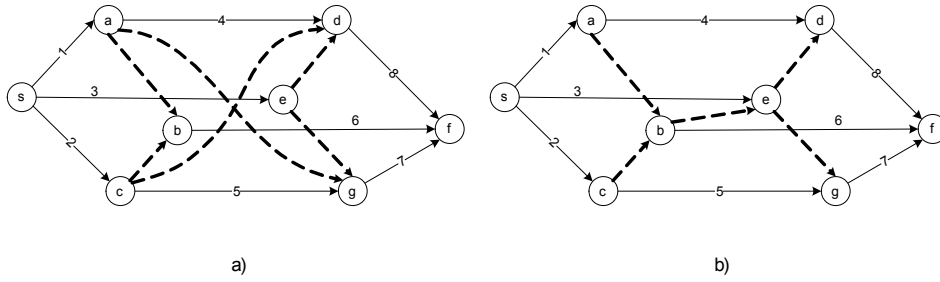
2. yapay kenar ekleme kuralı, 1. yapay kenar ekleme kuralının simetriği olması nedeni ile “Yapay_Kenar_Ekleme_Kuralı_2” yordamının algoritma karmaşıklığı $O(n)$ ile üstten sınırlıdır.

3.1.3.3 3. Yapay Kenar Ekleme Kuralı

G çizgesindeki x ve y düğümleri ile ilgilenelim. x düğümünü baş kabul eden hiçbir gerçel aktivite olmasın($\bar{P}_x = \emptyset$) ve x düğümünü baş kabul eden en az iki yapay kenar olsun($|P_x| > 1$). y düğümünü kuyruk kabul eden hiçbir gerçel aktivite olmasın($\bar{S}_y = \emptyset$) ve y düğümünü kuyruk kabul eden en az iki yapay kenar olsun($|S_y| > 1$). x ve y düğümleri arasında yapay ya da gerçel aktivite yok ve x düğümünün her bir yakın öncelinden, y düğümünün tüm yakın ardıllarına bir

kenar(yapay ya da gerçel) var ve bunlardan en az iki tanesi yapay ise 3. yapay kenar ekleme kuralı uygulanır.

Şekil 3.8’de, 3. yapay kenar ekleme kuralının uygulanmasına ilişkin bir örnek verilmiştir. Şekil 3.8(a)’da verilen çizgedeki $b(=x)$ ve $e(=y)$ düğümleri ile ilgilenelim. b düğümünü baş kabul eden ve e düğümünü kuyruk kabul eden hiçbir gerçel aktivite bulunmamaktadır. b düğümünün tüm yakın öncellerinden, e düğümünün tüm yakın ardıllarına bir kenar vardır($\{<a,d>, <a,g>, <c,d>, <c,g>\} \in (E \cup D)$) ve bu kenarlardan ikisi yapay kenardır($\{<a,g>, <c,d>\} \in D$). b ve e düğümleri arasında yapay ya da gerçel aktivite olmadığından, bu düğümler üzerinde 3. yapay kenar ekleme kuralı uygulanır. Bu durumda, b düğümünün yakın öncellerinin kuyruk düğümlerini kuyruk kabul eden, e düğümünün yakın ardıllarının baş düğümlerini baş kabul eden yapay kenarlar çizgeden çıkarılır ve $<b,e>$ yapay kenarı çizgeye eklenir. Şekil 3.8(a)’da verilen çizgeye $<b,e>$ yapay kenarının eklenmesi sonucu oluşan çizge Şekil 3.8(b)’de verilmiştir.



Şekil 3.8 Bir çizge örneği(a) ve bu çizgedeki b ve e düğümleri arasında 3. yapay kenar ekleme kuralının uygulanması sonucu oluşan çizge(b)

3. yapay kenar ekleme kuralına ilişkin yordam aşağıda verilmiştir :

```

Yapay_Kenar_Ekleme_Kuralı_3(x,y)
{
  if (( $\bar{p}_x = \bar{s}_y = \emptyset$ )  $\wedge$  ( $<x,y> \notin D$ )  $\wedge$  ( $<y,x> \notin D$ )  $\wedge$  ( $|F_x| > 1 \wedge |F_y| > 1$ ))
  {
    if ( $\forall p \forall s [ <p,s> \in D \vee <p,s> \in E: p \in F_x, s \in F_y ]$ ) // x düğümünün her bir yakın öncelinden y düğümünün tüm
    {
       $F = \{ <p,s> : <p,s> \in D, \text{ for each } p \in F_x, s \in F_y \}$  // yakın ardıllarına bir kenar var mı?
      if ( $|F| > 1$ ) // en az iki yapay kenar varsa,
      {
         $D = D - F$  // bu kenarları çizgeden çıkart
         $D = D \cup \{ <x,y> \}$ 
      }
    }
  }
}

```

$|S_x|$ ve $|S_y|$ değerleri en fazla $O(n)$ olabileceğinden, yordamdaki ikinci “if” komutunun işletim süresi $O(n^2)$ ’i aşmaz. Dolayısıyla, “Yapay_Kenar_Ekleme_Kuralı_3” yordamının algoritma karmaşıklığı $O(n^2)$ ile üstten sınırlıdır.

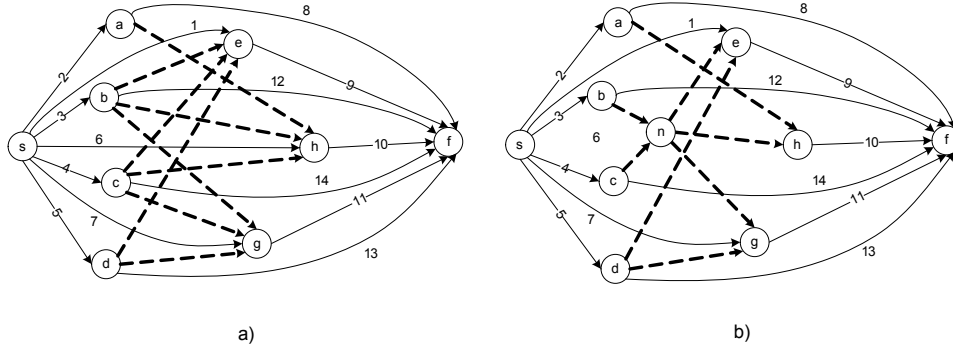
3.1.4 Düğüm Ekleme Kuralları

G çizgesindeki x ve y düğümleri ile ilgilenelim. x ve y düğümleri arasında gerçel ya da yapay kenar olmasın. Bu sınıfa giren kurallar, çizgeye yeni bir n düğümünün eklenmesi ile x , y düğümlerinden çıkan veya x , y düğümlerine gelen ortak yapay kenarların n düğümü ile ilişkilendirilmesi temel mantığına dayanır. x ve y düğümlerini kuyruk/baş kabul eden yapay kenarların baş/kuyruk düğümlerinden en az üç tanesi ortak ise düğüm ekleme kuralı uygulanır. Bu kural ile, x ve y düğümlerini kuyruk/baş kabul eden ortak yapay kenarlar çizgeden çıkarılır, x, y düğümleri ve ortak yapay kenarlar n düğümü ile ilişkilendirilir.

3.1.4.1 1. Düğüm Ekleme Kuralı

G çizgesindeki x ve y düğümleri ile ilgilenelim. x ve y düğümleri arasında yapay ya da gerçel kenar olmasın. x ve y düğümlerini kuyruk ya da/ve baş kabul eden gerçel kenarların olup olmaması bu kuralın uygulanmasında etkili değildir. x düğümünü kuyruk kabul eden yapay kenarların baş düğümlerinden en az üçü, y düğümünü kuyruk kabul eden yapay kenarların baş düğümleri ile aynı ise 1. düğüm ekleme kuralı uygulanır.

Şekil 3.9’da, 1. düğüm ekleme kuralının uygulanmasına ilişkin bir örnek verilmiştir. Şekil 3.9(a)’da verilen çizgedeki $b(=x)$ ve $c(=y)$ düğümleri ile ilgilenelim. b ve c düğümlerini kuyruk kabul eden yapay kenarların baş düğümleri ($S_b = \{e, h, g\}$, $S_c = \{e, h, g\}$) birbirine eşittir. Ortak olan en az üç yapay kenar olduğundan 1. düğüm ekleme kuralı uygulanır. Bu durumda, b ve c düğümlerinden ortak baş düğümlere giden yapay kenarlar çizgeden çıkarılır ve yeni bir n düğümü çizgeye eklenerek, n düğümünü kuyruk, ortak baş düğümleri baş düğüm kabul eden yapay kenarlar çizgeye eklenir. Öncelik ilişkilerini korumak için b ve c düğümlerinden n düğümüne birer yapay kenar kenar eklenir. Şekil 3.9(a)’da verilen çizgeye yeni bir düğüm eklenmesi sonucu oluşan çizge Şekil 3.9(b)’de verilmiştir.



Şekil 3.9 [20] no'lu çalışmadan alınan bir çizge örneği(a) ve bu çizgedeki b ve c düğümleri üzerinde 1. düğüm ekleme kuralının uygulanması sonucu oluşan çizge(b)

1. düğüm ekleme kuralına ilişkin yordam aşağıda verilmiştir :

```

Düğüm_Ekleme_Kuralı_1( $x, y$ )
{
  if(( $\langle x, y \rangle \notin D \wedge \langle y, x \rangle \notin D \wedge \langle x, y \rangle \notin E \wedge \langle y, x \rangle \notin E$ )  $\wedge$  ( $|\mathcal{S}_x \cap \mathcal{S}_y| > 2$ )) {
    create a new node  $n$            // yeni bir düğüm oluştur
     $V = V \cup \{ n \}$ 
     $D = D \cup \{ \langle x, n \rangle \}$        // kuyruğu  $x$ , baş düğümü  $n$  olan yapay kenar ekle
     $D = D \cup \{ \langle y, n \rangle \}$      // kuyruğu  $y$ , baş düğümü  $n$  olan yapay kenar ekle
    for (each  $z \in \mathcal{S}_x \cap \mathcal{S}_y$ )   // baş düğümü  $z$  -  $x$  ve  $y$  düğümlerinden çıkan kenarların baş düğümleri ortak olanlar-
       $D = D - \{ \langle x, z \rangle \}$      // ve kuyruk düğümü  $x$  olan yapay kenarı çizgeden çıkart
       $D = D - \{ \langle y, z \rangle \}$      // ve kuyruk düğümü  $y$  olan yapay kenarı çizgeden çıkart
       $D = D \cup \{ \langle n, z \rangle \}$    //  $n$  düğümünden  $z$  düğümüne yapay kenar ekle
  }
}

```

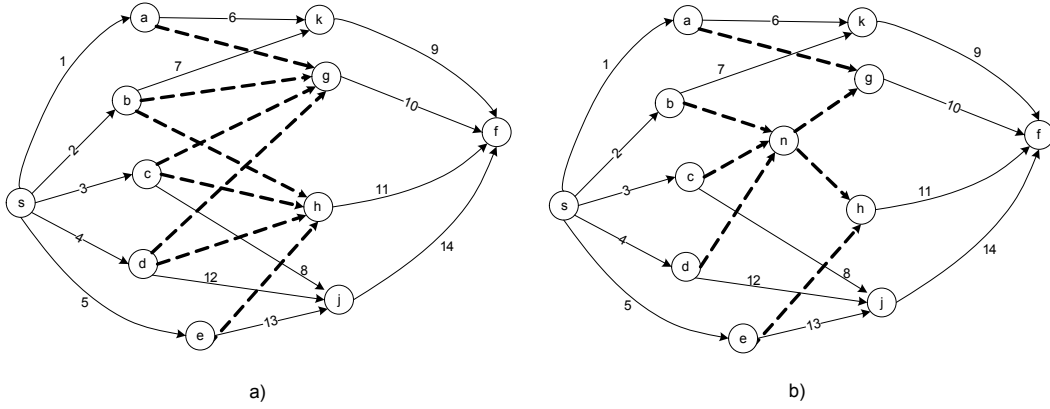
Yordamda yer alan “if” komutunun karmaşıklığı, $\mathcal{S}_x$ ve $\mathcal{S}_y$ kümeleri en fazla n eleman içerebileceğinden, $O(n)$ ile sınırlıdır. Yordamdaki “for” döngüleri, “if” komutundaki denetimden bağımsızdır. Dolayısıyla, “Düğüm_Ekleme_Kuralı_1” yordamının algoritma karmaşıklığı $O(n)$ 'dir.

3.1.4.2 2. Düğüm Ekleme Kuralı

G çizgesindeki x ve y düğümleri ile ilgilenelim. x ve y düğümleri arasında yapay ya da gerçel kenar olmasın. x ve y düğümlerini kuyruk ya da/ve baş kabul eden gerçel kenarların olup olmaması bu kuralın uygulanmasında etkili değildir. x düğümünü baş kabul eden yapay kenarların kuyruk düğümlerinden en az üçü, y düğümünü baş kabul eden yapay kenarların kuyruk düğümleri ile aynı ise 2. düğüm ekleme kuralı uygulanır.

Şekil 3.10'da, 2. düğüm ekleme kuralının uygulanmasına ilişkin bir örnek verilmiştir. Şekil 3.10(a)'da verilen çizgedeki g ve h düğümleri ile ilgilenelim. $g(=x)$

ve $h(=y)$ düğümlerini baş kabul eden yapay kenarların kuyruk düğümlerinden ($S_g=\{a,b,c,d\}$, $S_h=\{b, c, d, e\}$) üçü ortak olduğundan, 2. düğüm ekleme kuralı uygulanır. Bu durumda, ortak kuyruk düğümlerinden g ve h düğümlerine giden yapay kenarlar çizgeden çıkarılır ve yeni bir n düğümü çizgeye eklenerek, ortak kuyruk düğümlerinden n düğümüne giden yapay kenarlar eklenir. Öncelik ilişkilerini korumak için n düğümünden g ve h düğümlerine birer yapay kenar kenar eklenir. Şekil 3.10(a)'da verilen çizgeye yeni bir düğüm eklenmesi sonucu oluşan çizge Şekil 3.10(b)'de verilmiştir.



Şekil 3.10 Bir çizge örneği(a) ve bu çizgedeki g ve h düğümleri üzerinde 2. düğüm ekleme kuralının uygulanması sonucu oluşan çizge(b)

2. düğüm ekleme kuralına ilişkin yordam aşağıda verilmiştir :

```

Düğüm_Ekleme_Kuralı_2(x,y)
{
  if (( $\langle x,y \rangle \notin D \wedge \langle y,x \rangle \notin D \wedge \langle x,y \rangle \notin E \wedge \langle y,x \rangle \notin E$ )  $\wedge$  ( $|S_x \cap S_y| > 2$ ))
    create a new node  $n$            // yeni bir düğüm oluştur
     $V = V \cup \{n\}$ 
     $D = D \cup \{\langle n, x \rangle\}$          // kuyruğu  $n$ , baş düğümü  $x$  olan yapay kenar ekle
     $D = D \cup \{\langle n, y \rangle\}$          // kuyruğu  $n$ , baş düğümü  $y$  olan yapay kenar ekle.
    for (each  $z \in S_x \cap S_y$ )      // kuyruk düğümü  $z$  -  $x$  ve  $y$  düğümlerine gelen kenarlardan kuyruk düğümleri ortak olanlar-
       $D = D - \{\langle z, x \rangle\}$          // ve baş düğümü  $x$  olan yapay kenarı çizgeden çıkart
       $D = D - \{\langle z, y \rangle\}$          // ve baş düğümü  $y$  olan yapay kenarı çizgeden çıkart
       $D = D \cup \{\langle z, n \rangle\}$        // ve baş düğümü  $n$  olan yapay kenarı ekle
}

```

2. düğüm ekleme kuralı, 1. düğüm ekleme kuralının simetriği olması nedeni ile “Düğüm_Ekleme_Kuralı_2” yordamının algoritma karmaşıklığı $O(n)$ ile üstten sınırlıdır.

3.2 Önerilen Yaklaşımın Etkili Gerçekleştirimi

Bu tez çalışmasında, önerilen yaklaşımın bilgisayar ortamında etkili bir şekilde gerçekleştirilmesi için indirgeme kurallarının nasıl ele alınacağı ve kullanılacak veri yapıları tanıtılmıştır.

3.2.1 İndirgeme Kurallarının Ele Alınması

İndirgeme kurallarının, KAA üzerinde etkili bir şekilde uygulanabilmesi için nasıl ele alınacağı bu tez çalışmasında verilmiştir. İndirgeme kurallarının, KAA üzerindeki hangi düğümler üzerinde ve hangi sırada uygulanacağı açıklanmıştır.

3.2.1.1 İndirgeme Listeleri

Önerilen yaklaşımın etkili gerçekleştirilmesi için aşağıdaki indirgeme listeleri tanımlanmıştır :

T : En az iki yapay aktivite ve en az bir gerçel aktivite çıkan kuyruk düğümlerin listesi. $T = \{x \mid |S_x| > 1 \wedge \bar{S}_x \neq \emptyset; x \in V\}$.

H : En az iki yapay aktivite ve en az bir gerçel aktivite gelen baş düğümlerin listesi. $H = \{x \mid |P_x| > 1 \wedge \bar{P}_x \neq \emptyset; x \in V\}$.

L_T : En az iki yapay aktivite çıkan ve hiçbir gerçel aktivite çıkmayan kuyruk düğümlerin listesi. $L_T = \{x \mid |S_x| > 1 \wedge \bar{S}_x = \emptyset; x \in V\}$

L_H : En az iki yapay aktivite gelen ve hiçbir gerçel aktivite gelmeyen baş düğümlerin listesi. $L_H = \{x \mid |P_x| > 1 \wedge \bar{P}_x = \emptyset; x \in V\}$

İndirgeme listelerini kullanmadaki amaç, indirgeme kurallarını anlamlı (x,y) ikilileri ile çağırarak ve böylece genel işletim performansını arttırmaktır. İndirgeme listelerinin, indirgeme kurallarına ilişkin yordamlarda günlenmesi ile, gereksiz (x,y) ikililerinin üretilmesi engellenmeye çalışılır.

İndirgeme listeleri ile bir KAA'daki hangi düğümler üzerinde hangi indirgeme kurallarının uygulanabileceği bilgisi elde edilir. İndirgeme kurallarına parametre olarak gönderilen x ve y düğümleri, kurala bağlı olarak ilgili indirgeme listelerinden seçilir. Örneğin, 1. örtüşme kuralına gönderilen x ve y düğümleri, L_T listesinden seçilir. Böylece 1. örtüşme kuralının uygulanması için gerekli olan, (x,y)

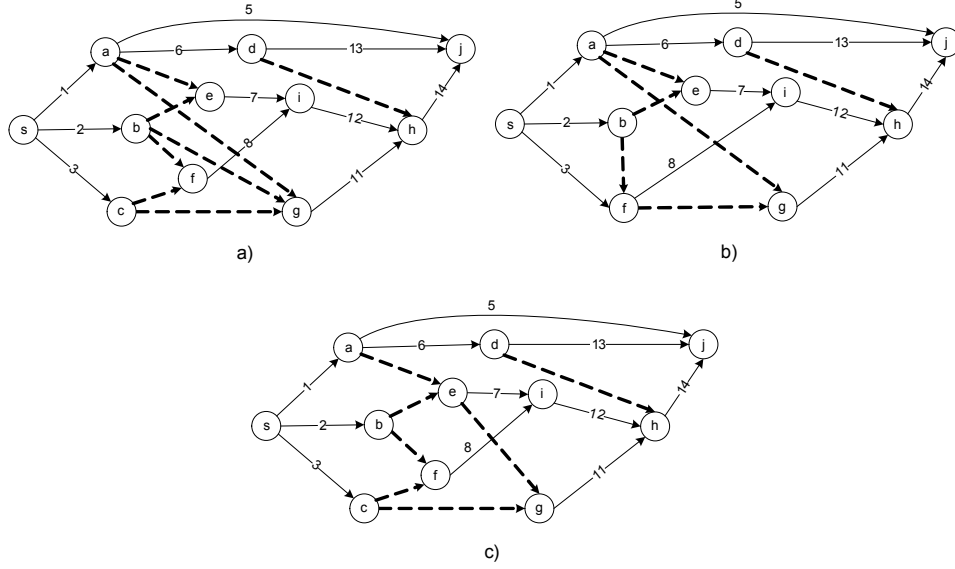
düğümlelerinden gerçel aktivite çıkmaması ve (x,y) düğümlerinden en az iki yapay aktivite çıkması koşulları sağlanmış olur. 1. örtüşme kuralı çizgedeki iki farklı düğüm üzerinde uygulandığından, L_T listesinde ikiden az eleman olması, 1. örtüşme kuralının uygulanamayacağını belirtir. Sonuç olarak, indirgeme listeleri ile çizgedeki her bir düğüm için, bu düğüm üzerinde her bir indirgeme kuralının uygulanıp uygulanamayacağı denetiminin yapılmasına gerek kalmaz.

3.2.1.2 İndirgeme Kurallarının İşletim Sırası

Önerilen yaklaşımın etkili gerçekleştirilmesi için indirgeme kurallarının uygulanacağı sıra aşağıda belirtilmiştir :

1. Yapay Kenar Büzme Kuralları
2. Düğüm Örtüşme Kuralları
3. Yapay Kenar Ekleme Kuralları
4. Düğüm Ekleme Kuralları

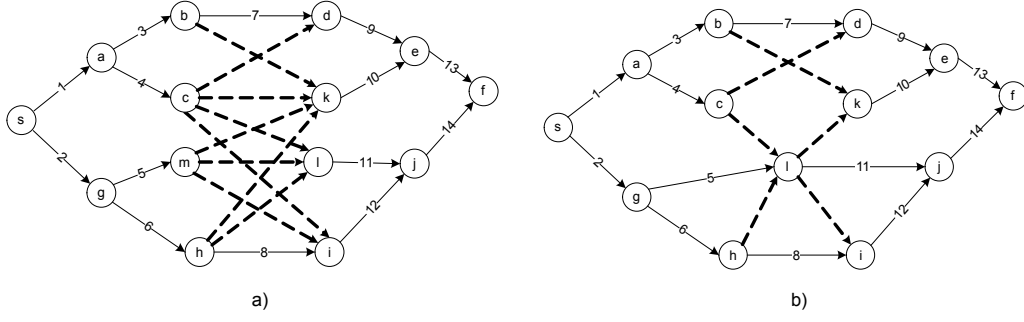
Bir KAA üzerinde, indirgeme kurallarının farklı sırada işletilmesi sonucu ortaya farklı çizgeler çıkabilmektedir. Şekil 3.11(a)'daki KAA üzerinde, indirgeme kuralları verilen sırada işletilirse, c ve f düğümleri üzerinde 1. yapay kenar büzme kuralının uygulanması sonucu 6 yapay kenar ve 10 düğüm içeren Şekil 3.11(b)'deki KAA elde edilir. Bu ağ üzerinde indirgeme kuralları farklı bir sırada işletilirse, e ve g düğümleri üzerinde 2. yapay kenar ekleme kuralının işletilmesi sonucu 7 yapay kenar ve 11 düğüm içeren Şekil 3.11(c)'deki KAA elde edilir. Şekil 3.11'deki örnekte görüldüğü gibi, indirgeme kurallarının önerilen sırada işletilmesi sonucu daha az yapay aktivite ve düğüm içeren KAA'lar elde edilir.



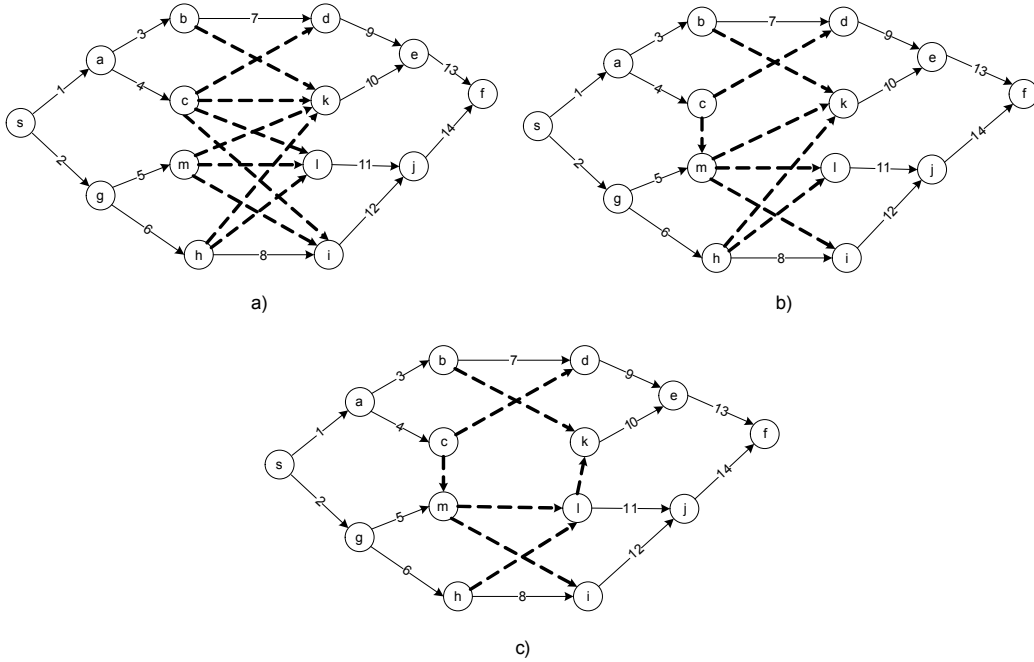
Şekil 3.11 [6] no'lu çalışmadan alınan örnek bir KAA(a) ve bu ağ üzerinde indirgeme kurallarının önerilen sırada(b) ve farklı sırada(c) uygulanması sonucu oluşan ağlar

İndirgeme kurallarının yukarıda önerilen sırada işletilmesi, KAA üzerinde yapılan işlemleri⁶ azaltmaktadır. Şekil 3.12(a)'daki KAA üzerinde indirgeme kuralları önerilen sırada işletilirse, m ve l düğümleri üzerinde 1. büzme kuralının işletilmesi sonucu Şekil 3.12(b)'deki KAA elde edilir. İndirgeme kuralları farklı bir sırada işletilirse, Şekil 3.12(b)'deki ağ elde edebilmek için 3 farklı indirgeme kuralı uygulanmalıdır. Şekil 3.13(a)'da yer alan ağda c ve m düğümleri üzerinde 1. yapay kenar ekleme kuralının işletilmesi sonucu Şekil 3.13(b)'deki ağ elde edilir. Bu ağ üzerindeki k ve l düğümleri üzerinde 2. yapay kenar ekleme kuralının uygulanması sonucu Şekil 3.13(c)'deki ağ elde edilir. Son olarak, Şekil 3.13(c)'deki ağda yer alan m ve l düğümleri üzerinde 1. büzme kuralının uygulanması sonucu Şekil 3.12(b)'deki ağ elde edilir.

⁶ İşlemler, KAA üzerinde indirgeme kurallarının uygulanmasını ve indirgeme kurallarına ilişkin yordamların farklı (x,y) ikilileri için çağrılmasını içerir.

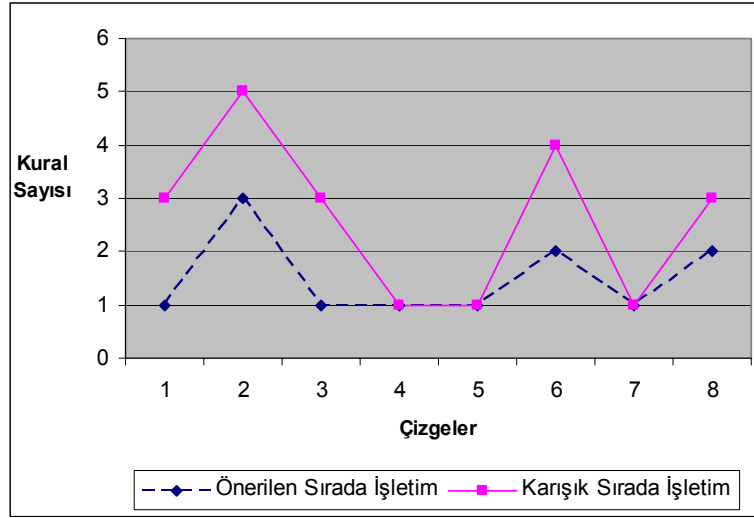


Şekil 3.12 [17] no'lu çalışmadan alınan örnek bir KAA(a) ve bu ağ üzerinde indirgeme kurallarının önerilen sırada uygulanması sonucu elde edilen KAA(b)

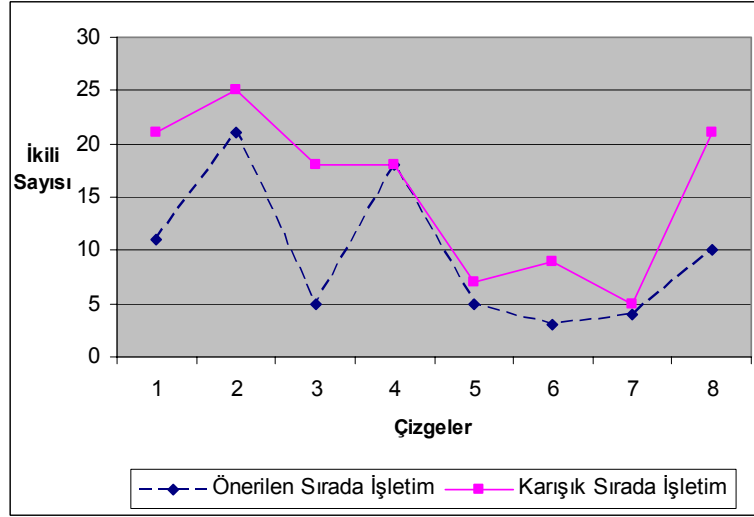


Şekil 3.13 [17] no'lu çalışmadan alınan örnek bir KAA(a) ve indirgeme kurallarının farklı sırada işletilmesi sonucu oluşan ağlar(b,c)

Şekil 3.14(a)'da, indirgeme kurallarının önerilen ve farklı sırada işletilmesi sonucu aynı çizgeyi elde edebilmek için çizge üzerinde uygulanan kural sayıları grafiksel olarak karşılaştırılmıştır. Şekil 3.14(b)'de ise, aynı çizgeler üzerinde, yordamların kaç farklı (x,y) ikilisi ile çağrıldığı karşılaştırılmıştır.



a)



b)

Şekil 3.14 Kuralların, farklı çizgeler üzerinde, önerilen ve farklı sırada uygulanmasının işletilen kural sayısına etkisi(a) ve üretilen (x,y) ikili sayısına etkisi(b)

3.2.1.3 Önerilen Algoritma

Bu tez çalışmasında, geliştirilen indirgeme kurallarının etkili gerçekleştirilmesi için bir algoritma önerilmiştir. Bu algoritma, ilgili indirgeme listelerinden seçilen (x,y) düğümleri için önerilen sırada indirgeme kurallarını işletir. Her bir indirgeme kuralına ilişkin yordamda, kuralın uygulanması sonucu çizgede meydana gelen değişiklikler indirgeme listelerine yansıtılır. Yapay kenar büzme kurallarının önerilen işletim sırasında en öncelikli kurallar olması ve 2. ile 3. yapay kenar büzme kurallarının basitliği nedeni ile, bir indirgeme kuralının uygulanması sonucu bu kuralların uygulanacağı bir durum meydana gelirse, bu kurallara ilişkin yordamlar (YapayKenarBüzmeKuralı2 ve YapayKenarBüzmeKuralı3) hemen çağırılır.

Şekil 3.15'te önerilen algoritmanın ana yordamı (İndirgeme_Kuralları) verilmiştir. İndirgeme_Kuralları yordamında, indirgeme listelerindeki düğümlerin indirgeme kuralları tarafından ele alınıp alınmadığını ifade etmek için aşağıdaki semboller kullanılmıştır :

I_T^+ = L_T listesinde yer alan ve indirgeme kuralları tarafından işlenmiş olan düğümler.

I_H^+ = L_H listesinde yer alan ve indirgeme kuralları tarafından işlenmiş olan düğümler.

T^+ = T listesinde yer alan ve indirgeme kuralları tarafından işlenmiş olan düğümler.

H^+ = H listesinde yer alan ve indirgeme kuralları tarafından işlenmiş olan düğümler.

I_T^- = L_T listesinde yer alan ve indirgeme kuralları tarafından işlenmemiş olan düğümler.

I_H^- = L_H listesinde yer alan ve indirgeme kuralları tarafından işlenmemiş olan düğümler.

T^- = T listesinde yer alan ve indirgeme kuralları tarafından işlenmemiş olan düğümler.

H^- = H listesinde yer alan ve indirgeme kuralları tarafından işlenmemiş olan düğümler.

İndirgeme_Kuralları yordamında, öncelikle KAA'daki 2. ve 3. yapay kenar büzme kuralları, uygulanma koşulunu sağlayan düğümler üzerinde uygulanır ve indirgeme listelerinin ilk değerleri atanır. İndirgeme listelerinde işlenecek düğümler olduğu sürece, önerilen sırada indirgeme kurallarına ilişkin $x_y_Üret$ yordamları çağırılır. Her bir indirgeme kuralının bir $x_y_Üret$ yordamı vardır. $x_y_Üret$ yordamında, kural ile ilgili listelerden (x,y) ikilileri seçilir ve kurala ilişkin yordam, bu ikililer ile çağırılır. Her bir indirgeme kuralına ilişkin $x_y_Üret$ yordamı, Şekil 3.16'da verilmiştir.

İndirgeme_Kuralları()

```

{
  for ( each  $x \in V$  )                                     // en öncelikli olarak 2. ve 3. büzme kurallarını uygula
    if(  $\bar{S}_x = \emptyset \wedge |\bar{S}_x| = 1$  )
      BüzmeKuralı2(x);
    if(  $\bar{P}_x = \emptyset \wedge |\bar{P}_x| = 1$  )
      BüzmeKuralı3(x);

   $T = \{x : |\bar{S}_x| \geq 1 \wedge |\bar{S}_x| > 1; x \in V\}$                // indirgeme listelerini oluştur
   $H = \{x : |\bar{P}_x| \geq 1 \wedge |\bar{P}_x| > 1; x \in V\}$ 
   $L_T = \{x : \bar{S}_x = \emptyset \wedge |\bar{S}_x| > 1; x \in V\}$ 
   $L_H = \{x : \bar{P}_x = \emptyset \wedge |\bar{P}_x| > 1; x \in V\}$ 
   $L_T^+ = L_H^+ = T^+ = H^+ = \emptyset$ 

   $L_T^- = L_T$ 
   $L_H^- = L_H$ 
   $T^- = T$ 
   $H^- = H$ 

  while (  $(L_T^- \neq \emptyset \vee L_H^- \neq \emptyset \vee T^- \neq \emptyset \vee H^- \neq \emptyset)$  ) { // indirgeme listelerinde, indirgeme kurallarında işlenmeyen düğümler var ise
    BüzmeKuralı1_x_y_Uret( $L_T$ );                             // ilgili indirgeme kuralı için gerekli x,y parametrelerini üret
    ÖrtüşmeKuralı1_x_y_Uret( $L_T^-, L_T^+$ );
    ÖrtüşmeKuralı2_x_y_Uret( $L_H^-, L_H^+$ );
    YapayKenarEklemeKuralı1_x_y_Uret( $L_T^-, L_T^+, T^-, T^+$ );
    YapayKenarEklemeKuralı2_x_y_Uret( $L_H^-, L_H^+, H^-, H^+$ );
    YapayKenarEklemeKuralı3_x_y_Uret( $L_T^-, L_T^+, L_H^-, L_H^+$ );
    DüğümEklemeKuralı1_x_y_Uret( $L_T^- \cup T^-, L_T^+ \cup T^+$ );
    DüğümEklemeKuralı2_x_y_Uret( $L_H^- \cup H^-, L_H^+ \cup H^+$ );

     $L_T^+ = L_T^+ \cup L_T^-$                                      // listeleri güncelle
     $L_H^+ = L_H^+ \cup L_H^-$ 
     $T^+ = T^+ \cup T^-$ 
     $H^+ = H^+ \cup H^-$ 

     $L_T^- = L_T^- \cup L_T^+$ 
     $L_H^- = L_H^- \cup L_H^+$ 
     $T^- = T^- \cup T^+$ 
     $H^- = H^- \cup H^+$ 
  }
}

```

Şekil 3.15 İndirgeme kurallarının etkili gerçekleştirilmesi için önerilen algoritmanın ana yordamı

```

BüzmeKuralı1_x_y_Üret( $L_T$ ){ // 1. Büzme Kuralı için x ve y ikililerini üret ve yordamı çağır
    for (each  $x \in L_T$ )
        for (each  $y \in \mathcal{G}_x$ )
            BüzmeKuralı1( $x, y$ );
}

```

```

Örtüşme Kuralı1_x_y_Üret( $L_T^-, L_T^+$ ){ // 1. Örtüşme Kuralı için x ve y ikililerini üret ve yordamı çağır
     $X : \{1,2,3,... | L_T^- | \} \rightarrow L_T^-$ 
    for (  $i : 1..| L_T^- |$  )
        for (  $j : i+1.. | L_T^- |$  )
            ÖrtüşmeKuralı1( $X_i, X_j$ );
    for (each  $x \in L_T^+$  )
        for (each  $y \in L_T^-$  )
            ÖrtüşmeKuralı1( $x, y$ );
}

```

```

Örtüşme Kuralı2_x_y_Üret( $L_H^-, L_H^+$ ){ // 2. Örtüşme Kuralı için x ve y ikililerini üret ve yordamı çağır
     $X : \{1,2,3,... | L_H^- | \} \rightarrow L_H^-$ 
    for (  $i : 1..| L_H^- |$  )
        for (  $j : i+1.. | L_H^- |$  )
            ÖrtüşmeKuralı2( $X_i, X_j$ );
    for (each  $x \in L_H^+$  )
        for (each  $y \in L_H^-$  )
            ÖrtüşmeKuralı2( $x, y$ );
}

```

```

YapayKenarEklemeKuralı1_x_y_Üret( $L_T^-, L_T^+, T^-, T^+$ ){ // 1. Yapay Kenar Ekleme Kuralı için x ve y ikililerini üret, yordamı çağır
    for (each  $x \in (L_T^- \cup T^-)$ )
        for (each  $y \in L_T^+$  )
            YapayKenarEklemeKuralı1( $x, y$ );
     $X : \{1,2,3,... | L_T^- | \} \rightarrow L_T^-$ 
    for (  $i : 1..| L_T^- |$  )
        for (  $j : i+1.. | L_T^- |$  )
            YapayKenarEklemeKuralı1( $X_i, X_j$ );
    for (each  $x \in (T^- \cup T^+)$ )
        for (each  $y \in L_T^-$  )
            YapayKenarEklemeKuralı1( $x, y$ );
}

```

Şekil 3.16 İndirgeme kurallarına ilişkin yordamlara parametre olarak gönderilen (x,y) düğüm ikililerini üreten x_y_Üret Yordamları

```

YapayKenarEklemeKuralı2_x_y_Üret( $L_H^-, L_H^+, H^-, H^+$ ) { // 2. Yapay Kenar Ekleme Kuralı için x, y ikililerini
    üret, yordamı çağır

    for (each x  $\in L_H^-$ )
        for (each y  $\in (L_H^+ \cup H^+)$ )
            YapayKenarEklemeKuralı2(x, y);
    X : {1,2,3,... |  $L_H^-$  |}  $\rightarrow L_H^-$ 
    for ( i : 1..|  $L_H^-$  |)
        for (j : i+1..|  $L_H^-$  |)
            YapayKenarEklemeKuralı2(Xi, Xj);
    for (each y  $\in H^-$ )
        for (each x  $\in (L_H^- \cup L_H^+)$ )
            YapayKenarEklemeKuralı2(x, y);
}

```

```

YapayKenarEklemeKuralı3_x_y_Üret( $L_T^-, L_T^+, L_H^-, L_H^+$ ) { // 3. Yapay Kenar Ekleme Kuralı için x ve y ikililerini
    üret, yordamı çağır

    for (each x  $\in L_H^-$ )
        for (each y  $\in L_T^-$ )
            YapayKenarEklemeKuralı3(x, y);
    for (each x  $\in L_H^+$ )
        for (each y  $\in L_T^-$ )
            YapayKenarEklemeKuralı3(x, y);
    for (each y  $\in L_T^+$ )
        for (each x  $\in L_H^-$ )
            YapayKenarEklemeKuralı3(x, y);
}

```

```

DüğümEklemeKuralı1_x_y_Üret( $L_T^- \cup T^-, L_T^+ \cup T^+$ ) { // 1. Düğüm Ekleme Kuralı için x ve y ikililerini üret,
    yordamı çağır

    X : {1,2,3,... |  $L_T^- \cup T^-$  |}  $\rightarrow L_T^- \cup T^-$ 
    for ( i : 1..|  $L_T^- \cup T^-$  |)
        for (j : i+1..|  $L_T^- \cup T^-$  |)
            DüğümEklemeKuralı1(Xi, Xj);
    for (each x  $\in L_T^+ \cup T^+$ )
        for (each y  $\in L_T^- \cup T^-$ )
            DüğümEklemeKuralı1(x, y);
}

```

Şekil 3.16 İndirgeme kurallarına ilişkin yordamlara parametre olarak gönderilen (x,y) düğüm ikililerini üreten x_y_Üret Yordamları (devam ediyor)

```

DüğümEklemeKuralı2_x_y_Üret( $L_H^- U H^-$ ,  $L_H^+ U H^+$ ) { // 2. Düğüm Ekleme Kuralı için x ve y ikililerini üret,
yordamı çağır
  X : {1,2,3,... |  $L_H^- U H^-$  |}  $\rightarrow L_H^- U H^-$ 
  for ( i : 1..| $L_H^- U H^-$  |)
    for (j : i+1..| $L_H^- U H^-$  |)
      DüğümEklemeKuralı2( $X_i$ ,  $X_j$ );
  for (each x  $\in L_H^+ U H^+$  )
    for (each y  $\in L_H^- U H^-$  )
      DüğümEklemeKuralı2(x, y);
}

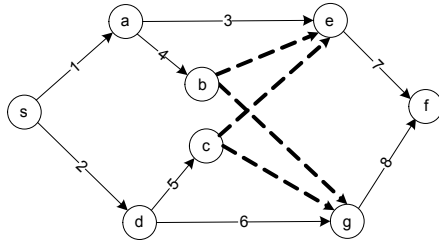
```

Şekil 3.16 İndirgeme kurallarına ilişkin yordamlara parametre olarak gönderilen (x,y) düğüm ikililerini üreten x_y_Üret Yordamları(devam ediyor)

Şekil 3.19-Şekil 3.28 arasında verilen her bir indirgeme kuralına ilişkin yordam, öncelikle kuralın uygulanıp uygulanamayacağına ilişkin denetimi yapar. Eğer kural uygulanır ise, kuralın uygulanması sonucu çizgede oluşan değişiklikler V , E , D ve indirgeme listelerine yansıtılır.

Şekil 3.17’de, örnek bir KAA ile bu ağa karşılık gelen V , E , D ve indirgeme listeleri verilmiştir. Bu ağdaki b ve c düğümleri için Şekil 3.19’da yer alan **ÖrtüşmeKuralı1(b,c)** yordamı çağırıldığında, ilk olarak yordamın 1. satırında b ve c düğümleri üzerinde 1. Örtüşme Kuralının uygulanıp uygulanamayacağı($S_b = S_c = \{e, g\}$) ve uygulama sonucu paralel kenar oluşup oluşmayacağı($\bar{P}_b \cap \bar{P}_c = \emptyset$) denetlenir. 3. satırda b düğümünden çıkan yapay kenarlar çizgeden çıkarılır($\{<b,e>, <b,g>\}$). Çıkarılan her bir yapay kenar için indirgeme listelerindeki değişiklikler denetlenir. Çıkarılan yapay kenarın kuyruk düğümü(b); kuralın uygulanması sonucu çizgeden, dolayısıyla L_T ve V listelerinden çıkarılır(16. ve 17. satır). Çıkarılan yapay kenarların baş düğümlerine gelen yapay kenar sayısına bakılır(4. satır). İlgili düğüme gelen yapay kenarlardan birinin silinmesi sonucu, bu düğüme bir tane yapay kenar geliyor ise, bu düğüme gelen gerçel kenar olup olmadığı denetlenir(5. satır). Düğüme gerçel kenar geliyor ise, düğüm H listesinden çıkarılır. Düğüme gerçel aktivite gelmiyor ise, düğüm L_H listesinden silinir ve YapayKenarBüzmeKuralı3 yordamı çağırılır. Şekil 3.17’deki çizgede $<b,e>$ ve $<b,g>$ yapay kenarlarının silinmesi ile e ve g düğümlerine gelen yapay kenar sayısının ikinin altına düşmesi sonucu, e ve g düğümleri H listesinden çıkarılır(9. satır). İndirgeme listelerindeki değişiklikler yapıldıktan sonra, b düğümüne gelen yapay ve gerçel aktiviteler c

düğümüne yönlendirilir(10.-17. satırlar). 1. Düğüm Örtüşme Kuralı uygulandıktan sonra çizgenin ve listelerin durumu Şekil 3.18'de verilmiştir.

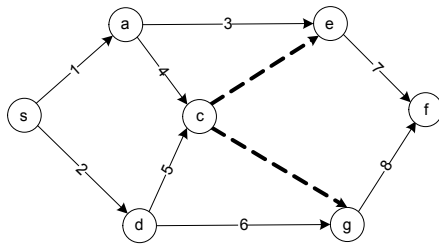


a)

$V = \{s, a, b, c, d, e, f, g\}$
 $E = \{<s, a>, <s, d>, <a, b>, <a, e>, <d, c>, <d, g>, <e, f>, <g, f>\}$
 $D = \{<b, e>, <b, g>, <c, e>, <c, g>\}$
 $T = \{\}$
 $H = \{e, g\}$
 $L_T = \{b, c\}$
 $L_H = \{\}$

b)

Şekil 3.17 Örnek bir KAA(a) ve bu ağ üzerinde indirgeme kuralları uygulanmadan önce V , E , D , indirgeme listeleri(b)



a)

$V = \{s, a, c, d, e, f, g\}$
 $E = \{<s, a>, <s, d>, <a, c>, <a, e>, <d, c>, <d, g>, <e, f>, <g, f>\}$
 $D = \{<c, e>, <c, g>\}$
 $T = \{\}$
 $H = \{\}$
 $L_T = \{c\}$
 $L_H = \{\}$

b)

Şekil 3.18 3.17(a)'daki ağ üzerinde indirgeme kuralları uygulandıktan sonra KAA(a) ve V , E , D , indirgeme listeleri(b)

ÖrtüşmeKuralı1(x, y) // $x \in L_T, y \in L_T$	
{ // $\bar{S}_x = \bar{S}_y = \phi \wedge \bar{S}_x > 1 \wedge \bar{S}_y > 1$ koşulları sağlanmış	
1. if ($\bar{S}_x = \bar{S}_y \wedge \bar{P}_x \cap \bar{P}_y = \phi$)	// paralellik denetimi yap
2. for (each $z \in \bar{S}_x$)	// x düğümünden çıkan yapay aktiviteleri çizgeden çıkart
3. $D = D - \{<x,z>\}$	
4. if ($ \bar{S}_z = 1$)	// z düğümüne çıkartma sonucu bir yapay aktivite geliyorsa
5. if ($\bar{P}_z = \phi$)	// z düğümüne gerçel aktivite gelmiyor ise L_H 'tan,
6. $L_H = L_H - \{z\}$	
7. BüzmeKuralı3(z)	
8. else	
9. $H = H - \{z\}$	// geliyor ise H'tan z yi çıkart
10. for (each $r \in \bar{P}_x$)	// x düğümüne gelen gerçel aktiviteleri
11. $E = E - \{<r,x>\}$	// y düğümüne yönlendir
12. $E = E \cup \{<r,y>\}$	
13. for (each $d \in \bar{S}_x$)	// x düğümüne gelen yapay aktiviteleri
14. $D = D - \{<d,x>\}$	// y düğümüne yönlendir
15. $D = D \cup \{<d,y>\}$	
16. $V = V - \{x\}$	// x düğümünü çizgeden ve L_T listesinden çıkart
17. $L_T = L_T - \{x\}$	
}	

Şekil 3.19 1. Düğüm Örtüşme Kuralına ilişkin yordam

ÖrtüşmeKuralı2(x, y) // $x \in L_H, y \in L_H$	
{	
if ($\bar{S}_x = \bar{S}_y \wedge \bar{S}_x \cap \bar{S}_y = \phi$)	
for (each $z \in \bar{S}_x$)	
$D = D - \{<z,x>\}$	
if ($ \bar{S}_z = 1$)	
if ($\bar{S}_z = \phi$)	
$L_T = L_T - \{z\}$	
BüzmeKuralı2(z)	
else	
$T = T - \{x\}$	
for (each $d \in \bar{S}_x$)	
$D = D - \{<x,d>\}$	
$D = D \cup \{<y,d>\}$	
for (each $r \in \bar{S}_x$)	
$E = E - \{<x,r>\}$	
$E = E \cup \{<y,r>\}$	
$V = V - \{x\}$	
$L_H = L_H - \{x\}$	
}	

Şekil 3.20 2. Düğüm Örtüşme Kuralına ilişkin yordam

```

YapayKenarBüzmeKuralı1( x , y ) // x ∈ LT, y ∈ Sx
{
    //  $\bar{S}_x = \phi \wedge |S_x| > 1 \wedge \langle x, y \rangle \in D$  koşulları sağlanmış
    if (  $\bar{p}_y = \phi \wedge K \neq \phi \wedge L \neq \phi$  )
        if (  $(\forall_l \forall_k [ \langle l, k \rangle \in D \vee \langle l, k \rangle \in E : l \in L, k \in K ])$  )
            for (each l ∈ L)
                for (each k ∈ K)
                    if (  $\langle l, k \rangle \in D$  )
                        // kartezyan çarpımda yer alan (l,k) ikilisi
                        D=D-{ $\langle l, k \rangle$ }
                        // yapay aktivite ise çizgeden çıkart
                        c.silindi[l] = true
                        g.silindi[k] = true

    if (  $|S_y| > 1$  )
        y_eleman_liste = true
        D = D-{ $\langle x, y \rangle$ }
        // ilgilenilen yapay aktiviteyi çizgeden çıkart
        for (each k ∈ K)
            // K kümesindeki düğümleri baş düğüm kabul
            D = D-{ $\langle x, k \rangle$ }
            // eden ve kuyruğu x olan aktivitelerin kuyruk
            D = D ∪ { $\langle y, k \rangle$ }
            // düğümlerini y düğümü yap
        for (each d ∈ Sx)
            // x düğümüne gelen yapay aktiviteleri
            D = D - { $\langle d, x \rangle$ }
            // y düğümüne yönlendir
            D = D ∪ { $\langle d, y \rangle$ }
        for (each r ∈  $\bar{p}_x$ )
            // x düğümüne gelen gerçel aktiviteleri
            E = E - { $\langle r, x \rangle$ }
            // y düğümüne yönlendir
            E = E ∪ { $\langle r, y \rangle$ }
        V = V - {x}
        // x düğümünü çizgeden ve LT listesinden çıkart
        LT = LT - {x}
        for (each l ∈ L)
            // silinen yapay aktivitelerin kuyruk düğümlerini kontrol et
            if (c.silindi[l])
                if (  $|S_x| = 1$  )
                    // sadece bir yapay aktivite çıkıyor
                    if (  $\bar{S}_l = \phi$  )
                        // ve hiç gerçel aktivite çıkmıyor ise
                        LT = LT - {l}
                        // düğümü LT listesinden çıkart
                        BüzmeKuralı2(l)
                    else
                        // gerçel aktivite çıkıyor ise
                        T = T - {l}
                        // düğümü T listesinden çıkart

        for (each k ∈ K)
            // silinen yapay aktivitelerin kuyruk düğümlerini kontrol et
            if (g.silindi[k] )
                if (  $|S_k| = 1$  )
                    // sadece bir yapay aktivite geliyor
                    if (  $\bar{p}_k = \phi$  )
                        // ve hiç gerçel aktivite gelmiyor ise
                        LH = LH - {k}
                        // düğümü LH listesinden çıkart
                        BüzmeKuralı3(k)
                    else
                        // düğümü H listesinden çıkart
                        H = H - {k}

    if (  $|S_y| = 1 \vee \bar{p}_y = \phi$  )
        // değişikliler sonucu y düğümüne bir yapay aktivite geliyor veya
        LH = LH - {y}
        // gerçel aktivite geliyor ise y düğümünü LH listesinden çıkart
    if (  $|S_y| \geq 2 \wedge !y\_eleman\_liste$  )
        // değişiklikler sonucu y düğümünden en az iki yapay aktivite çıkıyor ve
        if (  $\bar{S}_y = \phi$  )
            // y düğümü LT ya da T listelerine
            LT = LT ∪ {y}
            // daha önce eklenmemişse düğümü ilgili listeye ekle
        else
            T = T ∪ {y}
}

```

Şekil 3.21 1. Yapay Kenar Büzme Kuralına ilişkin yordam

YapayKenarBüzmeKuralı2(x) // $y \in \mathcal{N}_x$

```

{
  if (  $\bar{P}_x \cap \bar{P}_y = \emptyset$  ) // paralel aktivite oluşmaması için denetim yap
  if (  $|\mathcal{N}_y| \geq 2$  ) // y düğümüne en az iki yapay aktivite geliyor ise
    if (  $\bar{P}_y = \emptyset$  ) // y düğümüne gerçel gelmiyor ise , büzme sonucu
       $L_H = L_H - \{y\}$  // y'ye gerçel aktivite geleceğinden y'yi  $L_H$ 'tan çıkart
    else // y düğümüne gerçel geliyor ise y H listesinin elemanıdır
      y_eleman_H = true
   $D = D - \{<x,y>\}$  // ilgilenilen yapay aktiviteyi çizgeden çıkart
  for (each d  $\in \mathcal{N}_x$ ) // x düğümüne gelen yapay aktiviteleri
     $D = D - \{<d,x>\}$  // y düğümüne yönlendir
     $D = D \cup \{<d,y>\}$ 
  for (each r  $\in \bar{P}_x$  ) // x düğümüne gelen gerçel aktiviteleri
     $E = E - \{<r,x>\}$  // y düğümüne yönlendir
     $E = E \cup \{<r,y>\}$ 
  if (y_eleman_H  $\wedge |\mathcal{N}_y| < 2$  ) // değişiklikler sonucu y düğümüne ikiden az yapay aktivite geliyor
     $H = H - \{y\}$  // ve y düğümü H listesinin elemanı ise, listeden çıkart
  else if (!y_eleman_H  $\wedge |\mathcal{N}_y| \geq 2$  ) // değişiklikler sonucu y düğümüne en az iki yapay aktivite geliyor,
     $H = H \cup \{y\}$  // y düğümü H listesinin elemanı değil ise, listeye ekle
     $V = V - \{x\}$  // x düğümünü çizgeden çıkart
}

```

Şekil 3.22 2. Yapay Kenar Büzme Kuralına ilişkin yordam

YapayKenarBüzmeKuralı3(y) // $x \in \mathcal{N}_y$

```

{
  if (  $\bar{S}_x \cap \bar{S}_y = \emptyset$  ) // paralel aktivite oluşmaması için denetim yap
  if (  $|\mathcal{N}_x| \geq 2$  ) // x düğümünden en az iki yapay aktivite çıkıyorsa
    if (  $\bar{S}_x = \emptyset$  ) // x düğümünden gerçel çıkmıyor ise
       $L_T = L_T - \{x\}$  // büzme sonucu x'ten gerçel aktivite çıkacağından, x'i  $L_T$ 'den çıkart
    else // x düğümünden gerçel çıkıyor ise x, T listesinin elemanıdır
      y_eleman_T = true
   $D = D - \{<x,y>\}$  // ilgilenilen yapay aktiviteyi çizgeden çıkart
  for (each z  $\in \mathcal{N}_y$ ) // y düğümünden çıkan yapay aktivitelerin
     $D = D - \{<y,z>\}$  // kuyruk düğümlerini x düğümü yap
     $D = D \cup \{<x,z>\}$ 
  for (each s  $\in \bar{S}_y$  ) // y düğümünden çıkan gerçel aktivitelerin
     $E = E - \{<y,s>\}$  // kuyruk düğümlerini x düğümü yap
     $E = E \cup \{<x,s>\}$ 
  if (y_eleman_T  $\wedge |\mathcal{N}_x| < 2$  ) // değişiklikler sonucu x düğümünden ikiden az yapay aktivite çıkıyor
     $T = T - \{x\}$  // ve x düğümü T listesinin elemanı ise, listeden çıkart
  else if (!y_eleman_T  $\wedge |\mathcal{N}_x| \geq 2$  ) // değişiklikler sonucu x düğümünden en az iki yapay aktivite çıkıyor,
     $T = T \cup \{x\}$  // x düğümü T listesinin elemanı değil ise, listeye ekle
     $V = V - \{y\}$  // y düğümünü çizgeden çıkart
}

```

Şekil 3.23 3. Yapay Kenar Büzme Kuralına ilişkin yordam

```

YapayKenarEklemeKuralı1( x , y ) // x ∈ ( T U LT ) , y ∈ LT
{
    //  $\bar{S}_y = \phi \wedge |\bar{S}_y| > 1$  koşulları sağlanmış
    if ( <x,y> ∉ D ∧ <y,x> ∉ D ∧ <x,y> ∉ E ∧  $\bar{S}_y \subseteq \bar{S}_x$  )
        for (each z ∈  $\bar{S}_y$ )
            // x düğümünü kuyruk kabul eden ve
            D = D - {<x,z>}
            // baş düğümünü  $\bar{S}_y$  den alan yapay aktiviteleri çıkar
            if ( | $\bar{S}_x$ | = 1 )
                // z düğümüne çıkartma sonucu bir yapay aktivite geliyor,
                if (  $\bar{P}_z = \phi$  )
                    // z'ye gerçel aktivite gelmiyor ise z'yi LH'ten çıkart
                    LH = LH - {z}
                    BüzmeKuralı3(z)
                else
                    H = H - {z}
                    // z'ye gerçel aktivite geliyor ise H'tan çıkart
            D = D ∪ {<x,y>}
            // x düğümünden y düğümüne bir yapay aktivite ekle
            if ( | $\bar{S}_x$ | ≤ 1 )
                // x düğümünden çıkan yapay aktivite sayısı 2 den az
                if (  $\bar{S}_x \neq \phi$  )
                    // ve x düğümünden gerçel aktivite çıkıyor ise
                    T = T - {x}
                    // x düğümünü T listesinden çıkart
                else
                    // x düğümünden gerçel aktivite çıkmıyor ise
                    LT = LT - {x}
                    // x düğümünü LT listesinden çıkart
            if ( | $\bar{S}_y$ | = 2 )
                // <x,y> yapay aktivitesinin eklenmesi ile
                H = H U {y}
                // y düğümüne iki yapay aktivite geliyorsa y'yi H'a ekle
}

```

Şekil 3.24 1. Yapay Kenar Ekleme Kuralına ilişkin yordam

```

YapayKenarEklemeKuralı2( x , y ) // x ∈ LH, y ∈ ( H U LH )
{
    //  $\bar{P}_x = \phi \wedge |\bar{P}_x| > 1$  koşulları sağlanmış
    if ( <x,y> ∉ D ∧ <y,x> ∉ D ∧ <x,y> ∉ E ∧  $\bar{P}_x \subseteq \bar{P}_y$  )
        for (each z ∈  $\bar{P}_x$ )
            // y düğümünü baş kabul eden ve kuyruk düğümünü  $\bar{P}_x$  den alan
            D = D - {<z,y>}
            // yapay aktiviteleri çıkart
            if ( | $\bar{S}_z$ | = 1 )
                // z düğümünden çıkartma sonucu bir yapay aktivite çıkıyor ise
                if (  $\bar{S}_z = \phi$  )
                    // z düğümünden gerçel aktivite çıkmıyor ise z'yi LT,
                    LT = LT - {p}
                    // çıkıyor ise T listesinden çıkart
                    Büzme_Kuralı_2(z)
                else
                    T = T - {p}
            D = D ∪ {<x,y>}
            // x düğümünden y düğümüne bir yapay aktivite ekle
            if ( | $\bar{P}_y$ | ≤ 1 )
                // y düğümüne gelen yapay aktivite sayısı ikiden küçük ise
                if (  $\bar{P}_y \neq \phi$  )
                    // ve y düğümüne gerçel aktivite gelmiyor ise LT,
                    H = H - {y}
                    // geliyor ise y'yi T listesinden,
                else
                    LH = LH - {y}
            if ( | $\bar{P}_x$ | = 2 )
                // <x,y> 'nin eklenmesi ile x'ten çıkan yapay aktivite sayısı iki ise
                T = T U {x}
                // x düğümünü T listesine ekle
}

```

Şekil 3.25 2. Yapay Kenar Ekleme Kuralına ilişkin yordam

YapayKenarEklemeKuralı3(x , y) // $x \in L_H, y \in L_T$

```

{
    //  $\bar{P}_x = \bar{S}_y = \emptyset \wedge |\bar{P}_x| > 1 \wedge |\bar{S}_y| > 1$  koşulları sağlanmış
    if ( $\langle x, y \rangle \notin D \wedge \langle y, x \rangle \notin D$ )
        // x düğümünün her bir yakın öncülünden y düğümünün tüm ardıllarına bir aktivite var mı?
        if ( $\forall_p \forall_s [\langle p, s \rangle \in D \vee \langle p, s \rangle \in E : p \in \bar{P}_x, s \in \bar{S}_y]$ )
            F = { $\langle p, s \rangle : \langle p, s \rangle \in D$ , for each  $p \in \bar{P}_x, s \in \bar{S}_y$ }
            if (|F| > 1)
                // bu aktivitelerin en az ikisi yapay ise
                for (each  $\langle p, s \rangle \in F$ )
                    D = D - { $\langle p, s \rangle$ }
                    // yapay aktiviteleri çıkart
                    if (| $\bar{P}_s$ | = 1)
                        // s düğümüne çıkartma sonucu bir yapay aktivite geliyor ise
                        if ( $\bar{P}_s = \emptyset$ )
                            // ve bu düğümüne gerçel aktivite gelmiyor ise düğümü  $L_H$  listesinden,
                            // geliyor ise H listesinden çıkart
                             $L_H = L_H - \{s\}$ 
                            BüzmeKuralı3(s)
                        else
                            H = H - {s}
                    if (| $\bar{S}_p$ | = 1)
                        // p düğümünden çıkartma sonucu bir yapay aktivite çıkıyor ise
                        if ( $\bar{S}_p = \emptyset$ )
                            // ve bu düğümünden gerçel aktivite çıkmıyor ise düğümü  $L_T$ ,
                            // çıkıyor ise T listesinden çıkart
                             $L_T = L_T - \{p\}$ 
                            BüzmeKuralı2(p)
                        else
                            T = T - {p}
                    D = D  $\cup$  { $\langle x, y \rangle$ }
                    // x düğümünden y düğümüne bir yapay aktivite ekle
                    if (| $\bar{S}_x$ | = 2)
                        //  $\langle x, y \rangle$ 'nin eklenmesi ile x düğümünden
                        T = T  $\cup$  {x}
                        // iki yapay aktivite çıkıyor ise x düğümünü T listesine ekle
                    if (| $\bar{P}_x$ | = 2)
                        //  $\langle x, y \rangle$ 'nin eklenmesi ile y düğümüne
                        H = H  $\cup$  {y}
                        // iki yapay aktivite geliyor ise y düğümünü H listesine ekle
}

```

Şekil 3.26 3. Yapay Kenar Ekleme Kuralına ilişkin yordam

DüğümEklemeKuralı1(x, y) // x, y $\in$ (T U L_T)

```
{
  if (<x,y>  $\notin$  D  $\wedge$  <y,x>  $\notin$  D  $\wedge$  <x,y>  $\notin$  E  $\wedge$  <y,x>  $\notin$  E  $\wedge$  |Sx  $\cap$  Sy| > 2)
    create a new node n // yeni bir düğüm oluştur
    V = V U {n}
    D = D U {<x,n>} // x ve y düğümlerinden yeni düğüme birer yapay aktivite ekle
    D = D U {<y,n>}
    for (each z  $\in$  Sx  $\cap$  Sy) // x ve y düğümlerinden ortak düğümlere giden yapay aktiviteleri
      D = D - {<x,z>} // çizgeden çıkart
      D = D - {<y,z>}
      D = D U {<n,z>} // yeni düğümden ortak düğümlere giden yapay aktiviteler ekle
      if (|Sx| = 1) // ortak düğüme bir yapay aktivite geliyor ise H listesinden çıkart
        H = H - {z}
      if (|Sx| = 1) // çıkartmalar sonucu x'ten bir yapay aktivite çıkıyor (<x,n>) is
        if (Sx =  $\phi$ ) // ve x düğümünden gerçel aktivite çıkıyor ise x'i LT,
          LT = LT - {x} // çıkıyor ise T listesinden çıkart
          BüzmeKuralı2(x)
        else
          T = T - {x}
      if (|Sy| = 1) // çıkartmalar sonucu y'den bir yapay aktivite çıkıyor (<x,y>) ise
        if (Sy =  $\phi$ ) // ve y düğümünden gerçel aktivite çıkıyor ise y'i LT
          LT = LT - {y} // çıkıyor ise T listesinden çıkart
          BüzmeKuralı2(y)
        else
          T = T - {y}
    LH = LH U {n} // yeni düğüme hem yapay aktivite geliyor hem de çıkıyor olduğundan
    LT = LT U {n} // bu düğümü LT ve LH listelerine ekle
}
```

Şekil 3.27 1. Düğüm Ekleme Kuralına ilişkin yordam

```

DüğümEklemeKuralı2( x , y ) // x ∈ (H U LH), y ∈ (H U LH)
{
  if (<x,y> ∉ D ∧ <y,x> ∉ D ∧ <x,y> ∉ E ∧ <y,x> ∉ E ∧ | $\bar{P}_x \cap \bar{P}_y$ | > 2)
    create a new node n // yeni bir düğüm oluştur
    V = V U {n}
    D = D U {<n,x>} // yeni düğümünden x ve y düğümlerine birer yapay aktivite ekle
    D = D U {<n,y>}
    for (each z ∈  $\bar{P}_x \cap \bar{P}_y$ ) // x ve y düğümlerine ortak düğümlerden gelen yapay aktiviteleri
      D = D - {<z,x>} // çizgeden çıkart
      D = D - {<z,y>}
      D = D U {<z,n>} // ortak düğümlerden yeni düğüme giden yapay aktiviteler ekle
      if (| $\bar{S}_x$ | = 1) // ortak düğümünden bir yapay aktivite çıkıyor ise T listesinden çıkart
        T = T - {z}
    if (| $\bar{P}_x$ | = 1) // çıkartmalar sonucu x'e sadece bir yapay aktivite geliyor (<n,x>) ise
      if ( $\bar{P}_x = \emptyset$ ) // ve x düğümlerine gerçel aktivite gelmiyor ise x'i LH,
        LH = LH - {x} // geliyor ise H listesinden çıkart
        BüzmeKuralı3(x)
      else
        H = H - {x}
    if (| $\bar{P}_y$ | = 1) // çıkartmalar sonucu y'e sadece bir yapay aktivite geliyor (<n,y>) ise
      if ( $\bar{P}_y = \emptyset$ ) // ve y düğümlerine gerçel aktivite gelmiyor ise y'i LH,
        LH = LH - {y} // geliyor ise H listesinden çıkart
        BüzmeKuralı3(y)
      else
        H = H - {y}
    LH = LH U {n} // yeni düğüme hem yapay aktivite geliyor hem de çıkıyor olduğundan
    LT = LT U {n} // bu düğümü LT ve LH listelerine ekle
}

```

Şekil 3.28 2. Düğüm Ekleme Kuralına ilişkin yordam

3.2.2 Veri Yapıları

3.2.2.1 Ara Çizge Veri Yapısı

Ara çizge veri yapısı, DAA'nın ara çizgeye dönüştürülmesinde kullanılır ve ara çizgeyi tutar. Ara çizge veri yapısı, her bir aktiviteye ilişkin özelliklerin tutulduğu bir diziden(AÇVY) oluşur. Ara çizge veri yapısı, Şekil 3.29'da verilmiştir.

AÇVY

Aktivite Numarası	En Büyük Kuyruk Düzüm				En Büyük Baş Düzüm				Kuyruk Düzümü Olmayan Kenarların Sayısı				Baş Düzümü Olmayan Kenarların Sayısı				Aktiviteye İlişkin Kenarların Sayısı				Kuyruk Düzümü Olmayan Kenarların Listesi				Baş Düzümü Olmayan Kenarların Listesi				Aktiviteye İlişkin Kenarların Listesi			
	*	a	1	0	*	a	1	0	*	a	1	0	*	a	1	0	*	a	1	0	*	a	1	0	*	a	1	0	*	a	1	0
1	*	a	1	0	1	0	1																									
2	a	*	0	1	1	1																										
3	a	*	0	1	1	1																										
4	*	*	0	0	0	0																										
5	*	*	0	0	0	0																										
6	a	*	0	1	1	1																										
7	*	*	0	0	0	0																										

Kuyruk Düzüm

Baş Düzüm

Sonrakı Aktiviteye Gösterge

Şekil 3.29 Ara Çizge Veri Yapısı

Ara çizge veri yapısındaki aktivitelere, proje girdisindeki aktivite numarası ile erişilir. Her bir aktiviteye ilişkin veri yapısında yer alan alanlar ve açıklamaları aşağıda verilmiştir :

En Büyük Kuyruk Düzüm : Ara çizgede bir aktiviteye ilişkin yer alan kenarların kuyruk düğümlerinin en büyüğü.

En Büyük Baş Düzüm : Ara çizgede bir aktiviteye ilişkin yer alan kenarların baş düğümlerinin en büyüğü.

Kuyruk Düzümü Olmayan Kenarların Listesi : Ara çizgede bir aktiviteye ilişkin yer alan kenarlardan kuyruk düğümleri atanmamış kenarların listesi.

Kuyruk Düzümü Olmayan Kenarların Sayısı : Ara çizgede bir aktiviteye ilişkin yer alan kenarlardan kuyruk düğümleri atanmamış kenarların sayısı.

Baş Düzümü Olmayan Kenarların Listesi : Ara çizgede bir aktiviteye ilişkin yer alan kenarlardan baş düğümleri atanmamış kenarların listesi.

Baş Döğümü Olmayan Kenarların Sayısı : Ara çizgede bir aktiviteye ilişkin yer alan kenarlardan baş döğümü atanmamış kenarların sayısı.

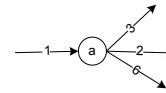
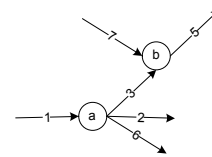
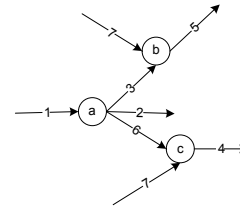
Aktiviteye İlişkin Kenarların Listesi : Ara çizgede bir aktiviteye ilişkin yer alan kenarların(kuyruk, baş döğümü atansın ya da atanmasın) listesi.

Aktiviteye İlişkin Kenarların Sayısı : Ara çizgede bir aktiviteye ilişkin yer alan kenarların(kuyruk, baş döğümü atansın ya da atanmasın) sayısı.

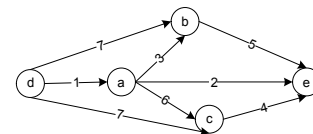
Önerilen ara çizge veri yapısının kullanımı ile ara çizgenin oluşturulmasına ilişkin yordamdaki(Şekil 2.2) temel işlemlerin(mantıksal karşılaştırma, bilgiye erişim) maliyetleri aşağıda örneklenmiştir :

1. “i no’lu aktivitenin ağda yer alıp almadığı” sorgulaması(6. ve 16. satırlar), AÇVY[i] ile erişilen aktivitenin ‘Aktiviteye İlişkin Kenarların Sayısı’ bilgisi ile $\Theta(1)$ ’de gerçekleştirilir.
2. Ara çizgede bulunan “i no’lu aktiviteye ilişkin tüm kenarların kuyruk ya da baş döğümlerinin atanıp atanmadığı” sorgulaması(9., 19., 32. ve 42. satırlar), AÇVY[i] ile erişilen aktivitenin ‘Aktiviteye İlişkin Kenarların Sayısı’ ile ‘Kuyruk Döğümü Olmayan Kenarların Sayısı’ ve ‘Baş Döğümü Olmayan Kenarların Sayısı’ bilgilerinin karşılaştırılması sonucu $\Theta(1)$ ’de gerçekleştirilir.
3. “i no’lu aktiviteye ilişkin kuyruk/baş döğümü atanmış kenarlardan kuyruk/baş döğümü en küçük olan kenarın kuyruk/baş döğümü” bilgisine(12. ve 22. satırlar), AÇVY[i] ile erişilen aktivitenin ‘Aktiviteye İlişkin Kenarlar Listesinin’ kuyruk ya da baş döğüm bilgisi ile $\Theta(1)$ ’de erişilir.
4. “i no’lu aktiviteye ilişkin kuyruk/baş döğümü atanmış kenarlardan kuyruk/baş döğümü en büyük olan kenarın kuyruk/baş döğümü” bilgisine(35. ve 45. satırlar), AÇVY[i] ile erişilen aktivitenin ‘En Büyük Kuyruk Döğümü’ ya da ‘En Büyük Baş Döğümü’ bilgisi ile $\Theta(1)$ ’de erişilir.

Şekil 2.1.(b)’de yer alan proje girdisi Generate_Intermediary_Graph yordamında işlenirken, oluşturulan ara çizge veri yapısı ve ara çizgenin değişimi Şekil 3.30’da verilmiştir.

[illegible][illegible]

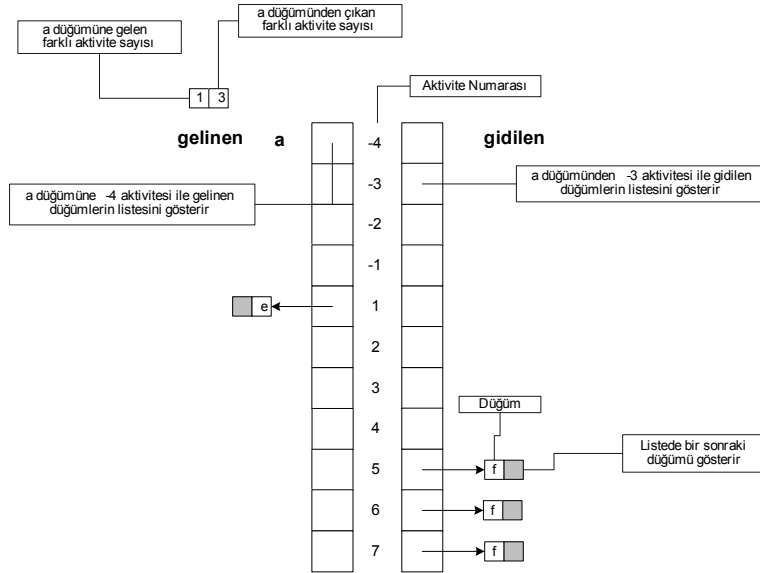
1	d	a	0	0	1				e a
2	a	e	0	0	1				a e
3	a	b	0	0	1				a b
4	c	e	0	0	1				c e
5	b	e	0	0	1				b e
6	a	c	0	0	1				a c
7	a	c	0	0	2				a b a c



60

3.2.2.2 Kenar Aktif Ağ Veri Yapısı

Kenar aktif ağ veri yapısı, ara çizgenin KAA'a dönüştürülmesinde kullanılır ve KAA'ı tutar. Kenar aktif ağ veri yapısında, her bir düğüm için gidilen ve gelinen olmak üzere iki dizi veri yapısı oluşturulur. Gidilen dizisi(gidilen) ile ilgili düğümden, hangi aktiviteler ile hangi düğümlere gidilir; gelinen dizisi(gelinen) ile hangi düğümlerden, hangi aktiviteler ile ilgili düğüme gelinir bilgisi elde edilir. Ayrıca, her bir düğüm için iki sayaç bilgisi tutulur. Bu sayaçlardan biri, ilgili düğümün kaç farklı aktivitenin kuyruk düğümü, diğeri ise ilgili düğümün kaç farklı aktivitenin baş düğümü olduğunu gösterir. Şekil 3.31'de bir KAA'daki bir a düğümünün veri yapısı verilmiştir.



Şekil 3.31 KAA'daki bir a düğümünün veri yapısı

Şekil 3.31'de görüldüğü gibi, gidilen ve gelinen dizilerinin boyu sabittir. Gidilen ve gelinen dizilerinin boyu; ara çizgede yer alan gerçel aktivitelerin sayısına, KAA'a eklenebilecek yapay aktivitelerin maksimum sayısı eklenerek hesaplanır. Ara çizgede k ($k > 1$) tane kenar ile ifade edilen bir aktivite, ara çizgeden KAA'a dönüşüm sırasında ağa en fazla $(k-1)$ yapay kenar eklenerek ifade edilir. Bu durumdan faydalanılarak, dizi boyu aşağıdaki biçimde hesaplanır :

$$K = \{K_i\}$$

K_i : ara çizgedeki i no'lu aktiviteye ilişkin kenarların sayısı

$$\text{Dizi boyu} : \text{gerçel aktivite sayısı} + \sum_{i \in V} (K_i - 1)$$

Şekil 3.32(b)'de yer alan ara çizgede 6 ve 7 numaralı aktivitelere ilişkin 3'er tane kenar olduğu görülmektedir. Bu ara çizgeden oluşturulacak KAA'a ait veri yapısındaki gidilen ve gelinen dizilerinin boyu aşağıdaki biçimde hesaplanır :

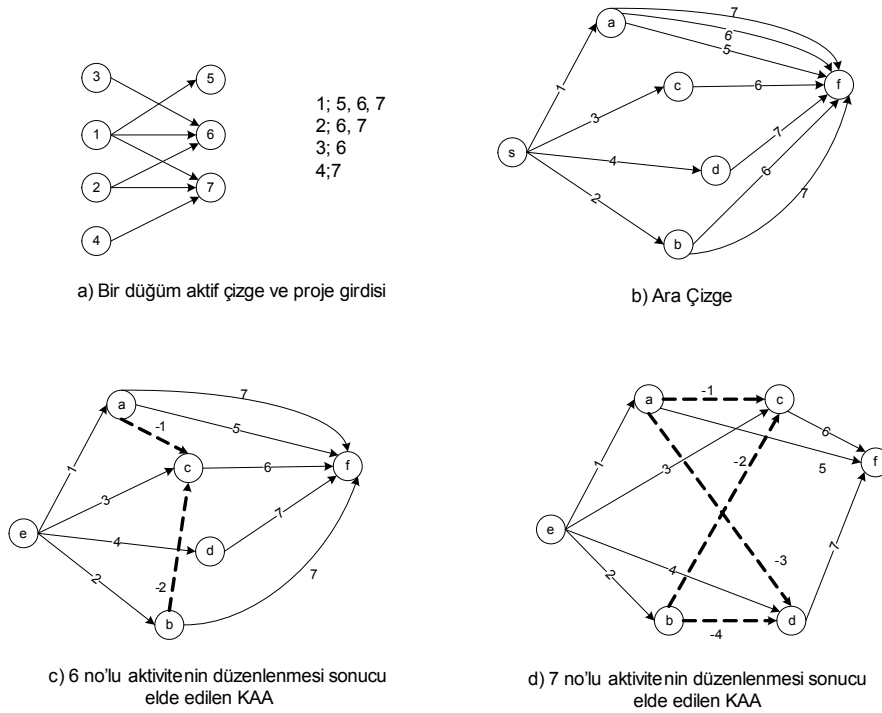
$$K = \{K_1, K_2, K_3, K_4, K_5, K_6, K_7\} = \{1, 1, 1, 1, 1, 3, 3\}$$

$$\text{dizi boyu} = 7 + 0 + 0 + 0 + 0 + 0 + (3-1) + (3-1) = 11$$

Önerilen kenar aktif ağ veri yapısının kullanımı ile kenar aktif ağın oluşturulmasına ilişkin yordamdaki(Şekil 2.3) temel işlemlerin maliyetleri aşağıda örneklenmiştir :

1. "i no'lu düğümden sadece j no'lu aktivitenin çıkıp çıkmadığı" sorgulaması (Tail_Node_Arrangement yordamındaki 6. satır), gidilen[i] ile erişilen 'düğümden çıkan farklı aktivite sayısı' bilgisi ile $\Theta(1)$ 'de gerçekleştirilir.
2. Tail_Node_Arrangement yordamının 3. satırındaki j no'lu aktiviteye ilişkin tüm kenarların silinmesi, T ve H listelerindeki her bir i düğümü için gidilen[i][j] ve gelinen[i][j] listeleri ile, $O(n)$ 'de gerçekleştirilir.

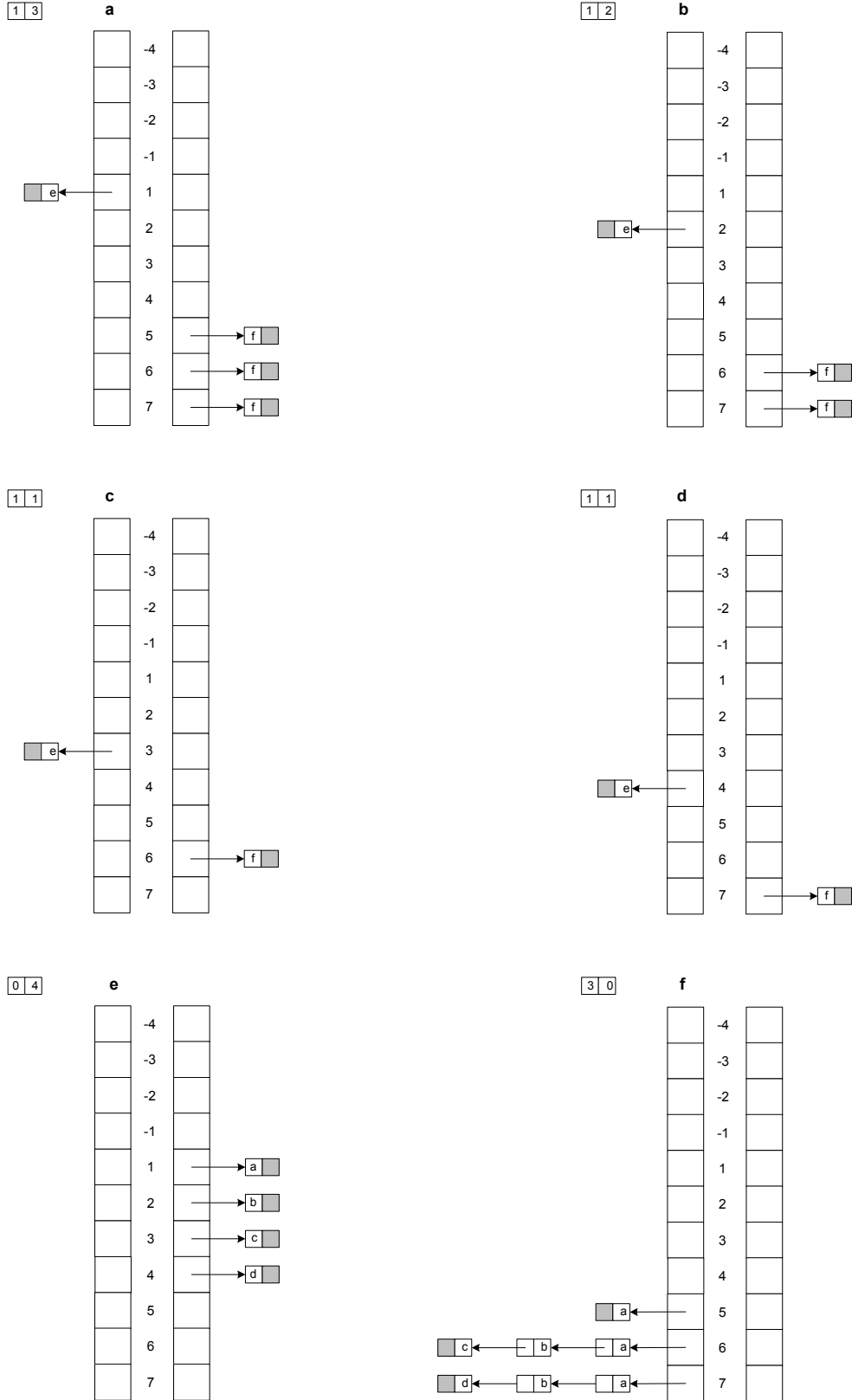
Bir DAA'ın KAA'a dönüşümü Şekil 3.32'de verilmiştir.



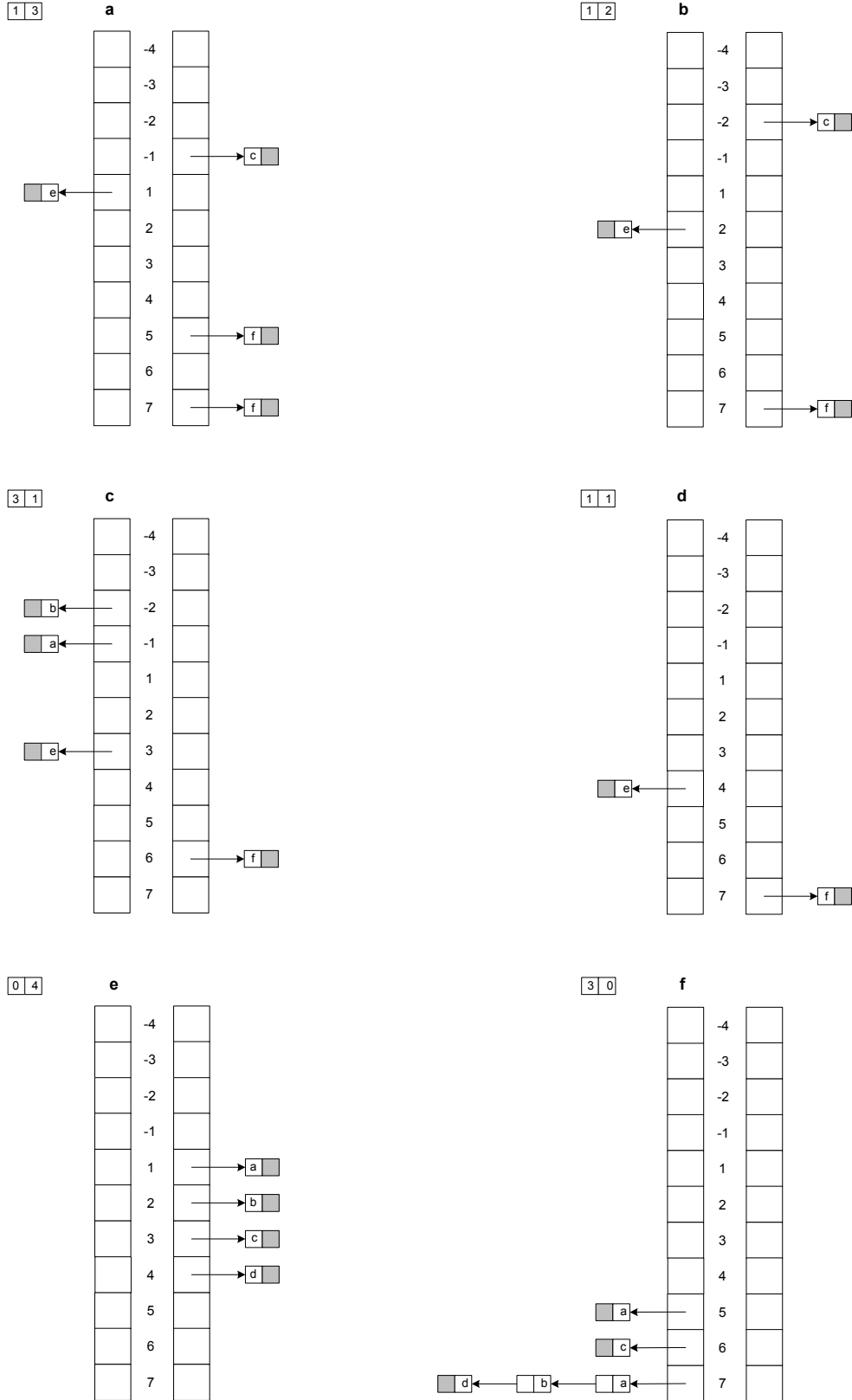
Şekil 3.32 Bir DAA'ın KAA'a dönüşümü

Şekil 3.32(a)'da verilen DAA, öncelikle ara çizgeye dönüştürülür. Şekil 3.32(b)'de yer alan ara çizgeyi tutan kenar aktif ağ veri yapısı Şekil 3.33'de verilmiştir. Şekil 3.32(b)'de yer alan ara çizgede 6 no'lu aktiviteye ait birden fazla kenar olduğu görülmektedir. 6 no'lu aktivitenin düzenlenmesi⁷ sonucu oluşan ağ Şekil 3.32(c)'de, bu ağa karşılık gelen veri yapısı Şekil 3.34'de verilmiştir. Şekil 3.32(c)'de yer alan çizgede 7 no'lu aktiviteye ait birden fazla kenar olduğu görülmektedir. 7 no'lu aktivitenin düzenlenmesi⁷ sonucu oluşan KAA Şekil 3.32(d)'de, bu ağa karşılık gelen veri yapısı Şekil 3.35'de verilmiştir.

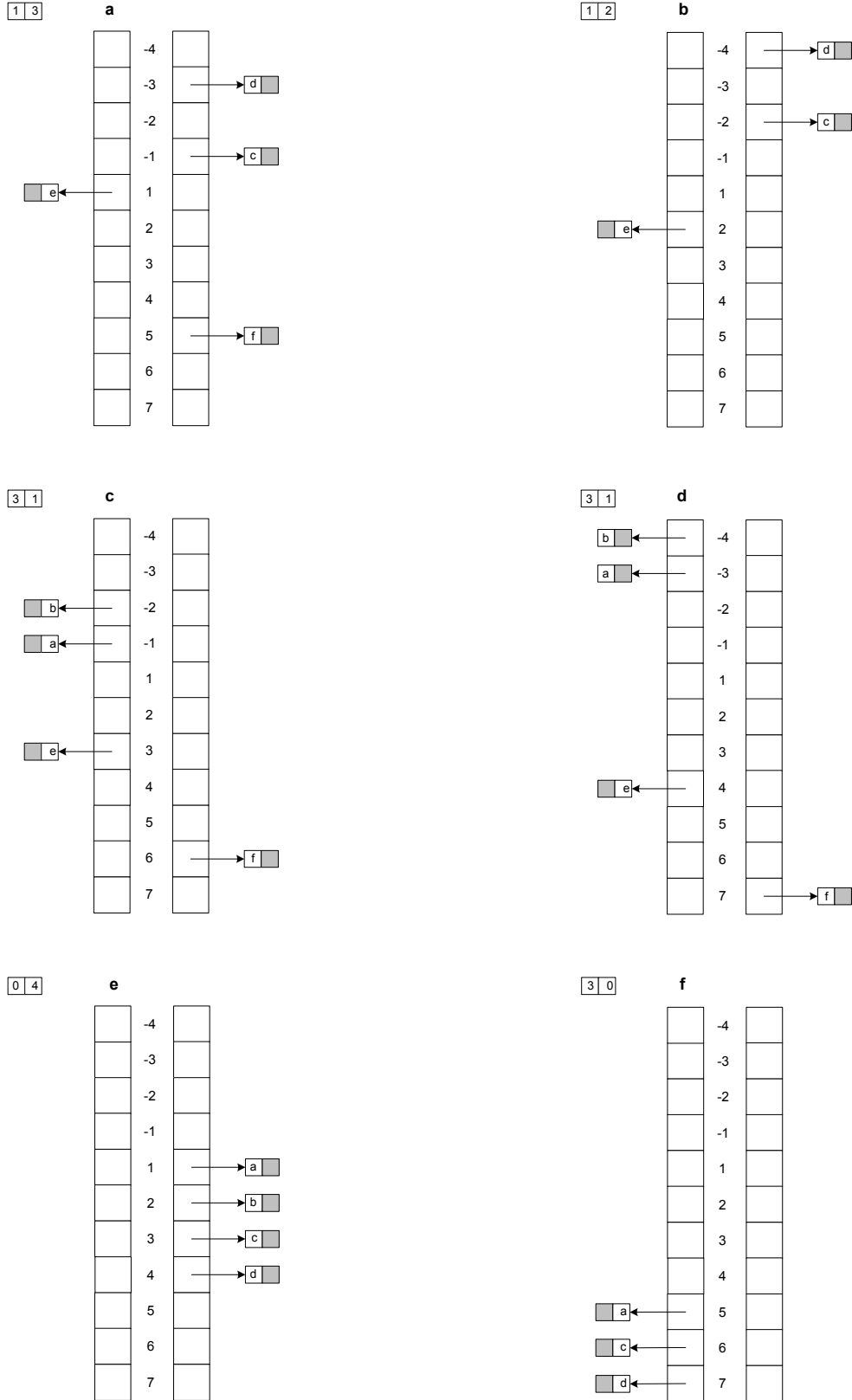
⁷ 6 ve 7 no'lu aktivitelerin kuyruk düğümleri Tail_Node_Arrangement yordamının 7. ile 14. satırlar arasında, baş düğümleri ise Head_Node_Arrangement yordamının 2. ve 3. satırlarında düzenlenir.



Şekil 3.33 Şekil 3.32(b)'deki ara çizgeye karşılık gelen KAA veri yapısı



Şekil 3.34 Şekil 3.32(c)'de yer alan ağa karşılık gelen KAA veri yapısı



Şekil 3.35 Şekil 3.32(d)'de yer alan ağa karşılık gelen KAA veri yapısı

3.2.2.3 Paralel Kenar Veri Yapısı

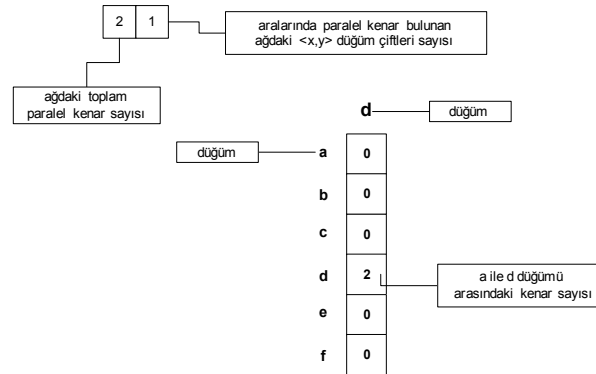
Paralel kenar veri yapısı, KAA'daki paralel kenarların sayısını hesaplamak ve paralel kenarları ayıklamak için oluşturulur.

3.2.2.3.1 Paralel Kenar Sayısı Veri Yapısı

Paralel kenar sayısı veri yapısı, KAA'daki paralel kenarların sayısını bulmak için oluşturulur. Bu veri yapısında her bir düğüm için bir dizi oluşturulur ve dizilerin her bir elemanı bir düğüme karşılık gelir. Bu diziler ile "ilgili düğümden hangi düğümlere kaç tane kenar ile gidilir" bilgisi elde edilir. Bu veri yapısında, ayrıca iki sayaç değeri bulunmaktadır. Bunlardan biri KAA'daki toplam paralel kenar sayısını, diğeri ise aralarında paralel kenarlar bulunan farklı $\langle x,y \rangle$ düğüm çiftlerinin sayısını gösterir. Bu sayaç değerleri kullanılarak, KAA'daki paralel kenarları ayıklamak için ağa kaç tane yapay aktivite(veya yapay düğüm) ekleneceği aşağıdaki biçimde hesaplanır:

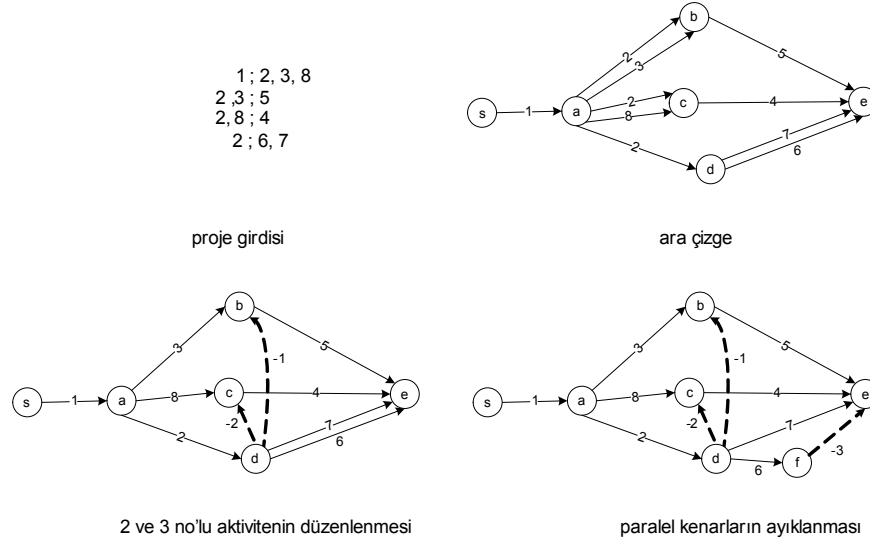
eklenecek yapay aktivite sayısı = $\frac{\text{toplam paralel kenar}}{\text{paralel kenar düğüm çiftleri}}$

Şekil 3.36'da bir KAA'daki bir d düğümü için paralel kenar sayısı veri yapısı verilmiştir.



Şekil 3.36 KAA'daki bir d düğümü için paralel kenar sayısı veri yapısı

Verilen bir proje girdisinin KAA'a dönüşümü Şekil 3.37'de verilmiştir.



Şekil 3.37 Proje girdisi(a), bu proje girdisine karşılık gelen ara çizge(b) ve KAA(c), paralel kenarların ayıklanması sonucu elde edilen KAA(d)

Şekil 3.37(a)'da verilen proje girdisinden yararlanılarak öncelikle ara çizge oluşturulur. Şekil 3.37(b)'deki ara çizgede yer alan paralel kenarların sayısını hesaplamak için oluşturulan veri yapısı Şekil 3.38(a)'da verilmiştir. Şekil 3.38(a)'daki veri yapısında ve Şekil 3.37(b)'deki ağda görüldüğü gibi, ağda 6 tane paralel kenar bulunmaktadır ve bu paralel kenarlar, 3 farklı düğüm çifti ($\langle a, b \rangle$, $\langle a, c \rangle$, $\langle a, d \rangle$) arasında yer almaktadır. Şekil 3.37(b)'deki ara çizge üzerinde gözlemlerin uygulanması sonucu oluşan KAA Şekil 3.37(c)'de, bu ağa karşılık gelen paralel kenar sayısı veri yapısı Şekil 3.38(b)'de görülmektedir. Şekil 3.38(b)'deki veri yapısında ve Şekil 3.37(c)'deki ağda görüldüğü gibi, ağda 2 tane paralel kenar bulunmaktadır ve bu paralel kenarlar, $\langle a, d \rangle$ düğüm çifti arasında yer almaktadır. Dolayısıyla, ağdaki paralel kenarları ayıklamak için ağa 1 tane yapay kenar eklenmelidir ($2-1=1$). Şekil 3.37(c)'deki paralel kenarların ayıklanması sonucu oluşan KAA, Şekil 3.37(d)'de verilmiştir.

a)

6	3
---	---

a

a	0
b	2
c	2
d	1
e	0
f	0

b

a	0
b	0
c	0
d	0
e	0
f	1

c

a	0
b	0
c	0
d	0
e	0
f	1

d

a	0
b	0
c	0
d	2
e	0
f	0

e

a	1
b	0
c	0
d	0
e	0
f	0

f

a	0
b	0
c	0
d	0
e	0
f	0

b)

2	1
---	---

a

a	0
b	1
c	1
d	1
e	0
f	0

b

a	0
b	0
c	0
d	1
e	0
f	1

c

a	0
b	0
c	0
d	1
e	0
f	1

d

a	0
b	0
c	0
d	2
e	0
f	0

e

a	1
b	0
c	0
d	0
e	0
f	0

f

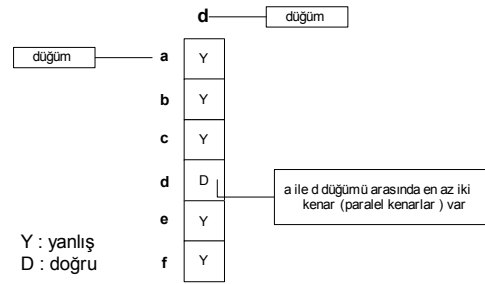
a	0
b	0
c	0
d	0
e	0
f	0

Şekil 3.38 Şekil 3.37(b)'de yer alan ara çizge için paralel kenar sayısı veri yapısı(a) ve Şekil 3.37(c)'de yer alan KAA için paralel kenar sayısı veri yapısı(b)

3.2.2.3.2 Paralel Kenar Veri Yapısı

Paralel kenar veri yapısı, KAA'da hangi düğümler arasında paralel kenar olduğunu bulmak ve bu kenarları ayıklamak için, KAA veri yapısından indirgeme veri yapısına geçerken oluşturulur. Bu veri yapısında her bir düğüm için bir dizi veri yapısı oluşturulur ve dizilerin her bir elemanı bir düğüme karşılık gelir. Bu dizi ile, ilgili düğüm ile hangi düğümler arasında paralel kenar var bilgisi elde edilir.

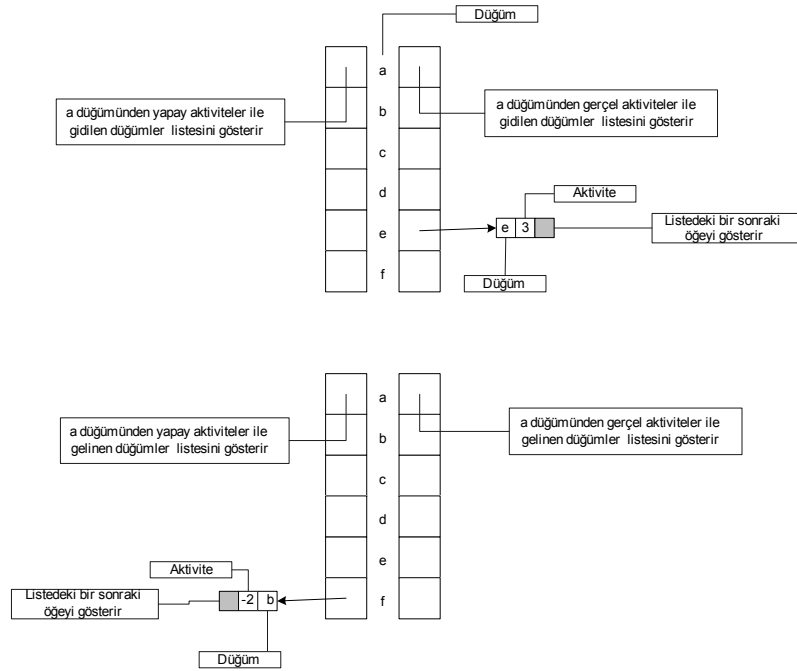
Şekil 3.39'da bir KAA'daki bir d düğümü için paralel kenar veri yapısı verilmiştir.



Şekil 3.39 KAA'daki bir d düğümü için paralel kenar veri yapısı

3.2.2.4 İndirgeme Veri Yapısı

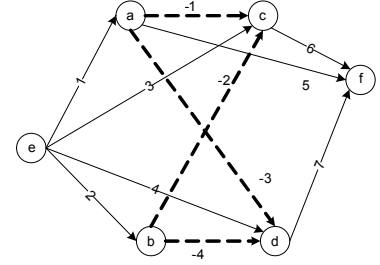
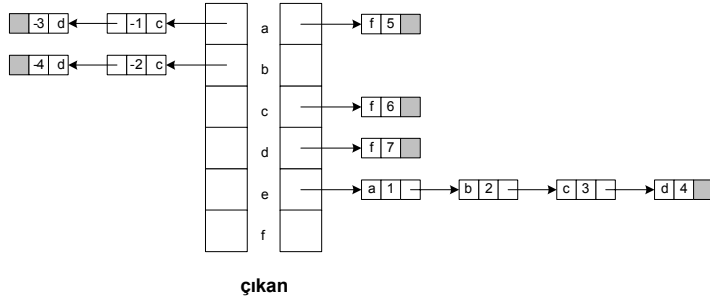
İndirgeme veri yapısı, KAA üzerinde indirgeme kurallarının uygulanması sırasında kullanılır ve KAA'ı tutar. İndirgeme veri yapısı, elemanları kenar aktif ağıdaki düğümler olan dört diziden oluşmaktadır. Dizilerin her bir elemanı bir listeye işaret eder. Her bir listede, aktivite numarası ve düğüm numarası bilgisinden oluşan öğeler bulunmaktadır. Dizilerden birincisi ile ilgili düğümden hangi gerçel aktiviteler ile hangi düğümlere gidilir, ikincisi ile hangi gerçel aktiviteler ile hangi düğümlerden ilgili düğüme gelinir bilgisi, üçüncüsü ile ilgili düğümden hangi yapay aktiviteler ile hangi düğümlere gidilir, dördüncüsü ile hangi yapay aktiviteler ile hangi düğümlerden ilgili düğüme gelinir bilgisi elde edilir. Şekil 3.40'ta bir KAA için indirgeme veri yapısı açıklanmıştır.



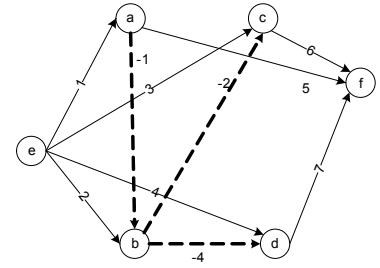
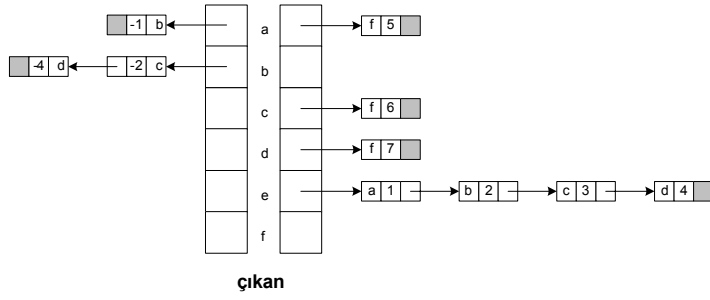
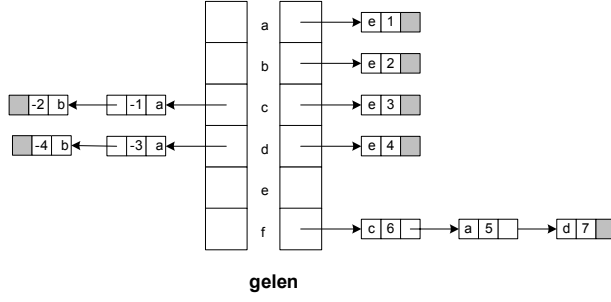
Şekil 3.40 İndirgeme veri yapısı

Şekil 3.40'da yer alan dizilerinin boyu sabittir. Dizilerin boyu; KAA'da yer alan aktivitelerin sayısına, paralel kenar sayısı veri yapısından yararlanılarak elde edilen -paralel kenarları ayıklamak için gerekli- yapay kenar sayısı eklenerek hesaplanır.

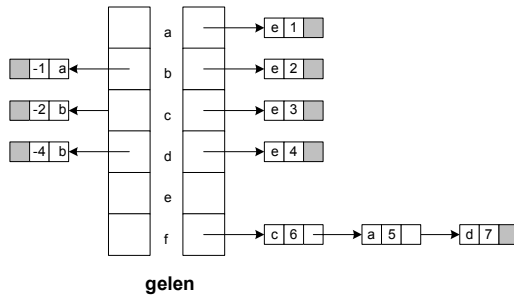
Şekil 3.41'de, [20] numaralı çalışmadan alınan bir KAA ve bu ağa karşılık gelen veri yapısı ile bu ağdaki a ve b düğümleri üzerinde 1. Yapay Kenar Ekleme Kuralının uygulanması sonucu ağın ve veri yapısının değişimi verilmiştir.



KAA



1. yapay kenar ekleme kuralı
uygulandıktan sonra
KAA



Şekil 3.41 [20] no'lu çalışmadan alınan bir KAA örneği ve bu ağa karşılık gelen indirgeme veri yapısı

3.3 Algoritma Analizi

DAA'dan KAA'a bir dönüşüm gerçekleştiren ön işleme algoritmasının karmaşıklığının $O(n^2)$ olduğu daha önce belirtilmiştir. İndirgeme kurallarına ilişkin yordamların analizleri ise ayrı ayrı ele alınmıştır. Gerçekte, indirgeme kurallarına ilişkin yordamların kaç kez çağrılacağı -indirgeme kurallarına ilişkin yordamların kaç farklı (x,y) ikilisi ile çağrılacağı- dikkate alındığında doğru bir analiz yapmak olasıdır. [7] numaralı çalışmada verilen en kötü durumda, (x,y) ikili sayısı "Yapay_Kenar_Büzme_Kuralı_1", "Yapay_Kenar_Ekleme_Kuralı_1", "Yapay_Kenar_Ekleme_Kuralı_2", "Yapay_Kenar_Ekleme_Kuralı_3", "Düğüm_Örtüşme_Kuralı_1", "Düğüm_Örtüşme_Kuralı_2", "Düğüm_Ekleme_Kuralı_1", "Düğüm_Ekleme_Kuralı_2" yordamları için en fazla $O(n^2)$; yalnızca bir yapay kenarın çıktığı (ya da geldiği) düğüm sayısı ile orantılı olması nedeni ile "Yapay_Kenar_Büzme_Kuralı_2", "Yapay_Kenar_Büzme_Kuralı_3" yordamları için de en fazla $O(n)$ olabilir. Bu durumda, verilen bir KAA'daki yapay kenarları azaltmak için indirgeme kurallarının uygulanmasının algoritma karmaşıklığı $O(n^4)$ ile üstten sınırlı olacaktır. Sonuç olarak, olabildiğince az yapay kenar sayılı KAA'ı oluşturmak için tezde önerilen yaklaşımın algoritma karmaşıklığı $O(n^4)$ ile üstten sınırlıdır[8].

4. ÖNERİLEN YAKLAŞIMIN DİĞER YAKLAŞIMLAR İLE KARŞILAŞTIRILMASI

Önerilen yaklaşım, [14] ve [18]'de verilen yaklaşımlar ile *algoritma karmaşıklığı*, *paralel kenar*, *geçişlilik ilişkisi*, *fazlalık aktivite* ve *yakın önceller* kriterlerine bakılarak karşılaştırılmıştır. Ayrıca; önerilen ve [14],[18]'de verilen yaklaşımların literatürden alınan düğüm aktif ağlar için oluşturdukları kenar aktif ağlar, yapay aktivite ve düğüm sayıları açılarından karşılaştırılmıştır..

Çizelge 4.1'de KAA oluşturmak için önerilen yaklaşım, son yıllarda bu konuda yapılmış çalışmalarda önerilen diğer yaklaşımlar ile karşılaştırılmıştır.

Çizelge 4.1 Kenar aktif ağ oluşturmada son yıllarda yapılmış olan çalışmaların karşılaştırılması

Yaklaşımlar	Algoritma Analizi	Paralel Kenarlar İçerir mi? ⁸	Geçişlilik İlişkisi İçerir mi? ⁸	Fazlalık Aktivite İçerir mi? ⁸	Yakın Öncel İlişkileri Bozulur mu? ⁹
Önerilen	$O(n^4)$	hayır	hayır	hayır	evet/hayır ¹⁰
Sharma	$O(n^2 m^2)$	evet	evet	evet	hayır
Michael ve diğ.	$\Omega(n^4)$	evet	hayır	hayır	evet

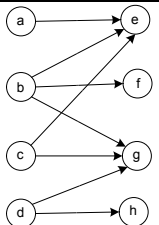
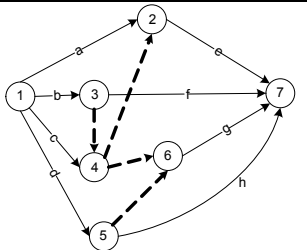
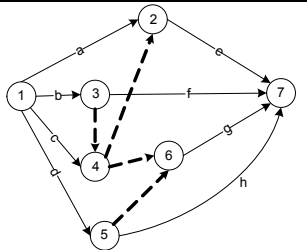
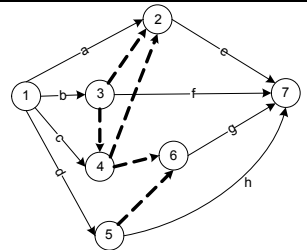
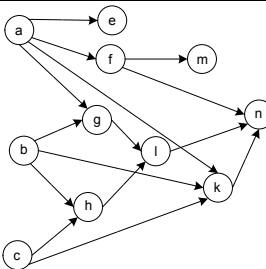
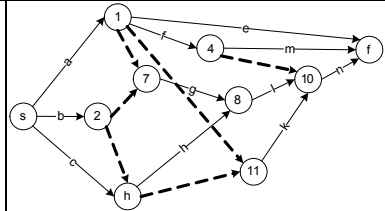
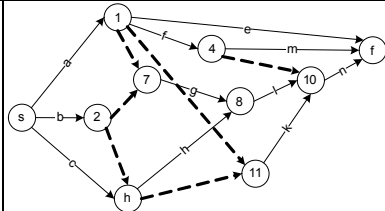
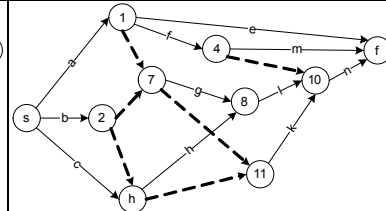
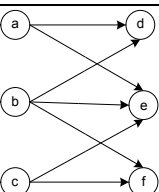
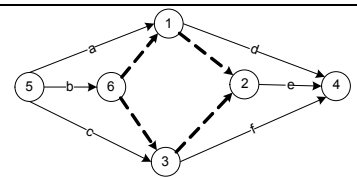
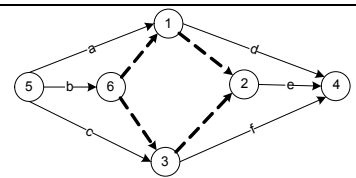
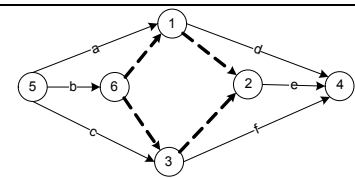
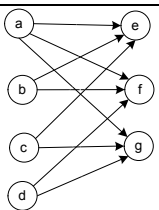
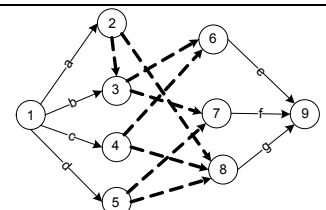
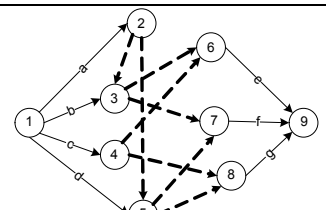
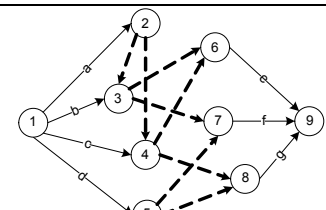
Çizelge 4.2'de literatürden alınan düğüm aktif ağ örnekleri için, önerilen yaklaşım ve diğer yaklaşımlar tarafından oluşturulan kenar aktif ağlar yapay aktivite sayısı ve düğüm sayısı açılarından karşılaştırılmıştır.

⁸ Proje kısıtlayıcılarından biri olması nedeni ile KAA'nın paralel kenarlar içermemesi, ağın gereksiz yapay kenarlar içermesine neden olduğundan KAA'nın geçişlilik ilişkisi ve fazlalık aktivite içermemesi tercih edilir.

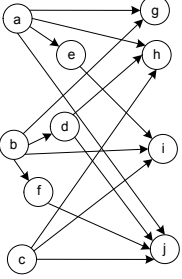
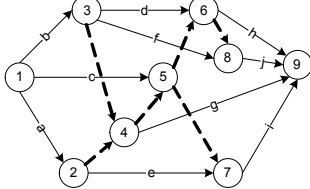
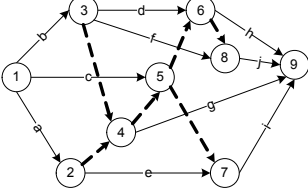
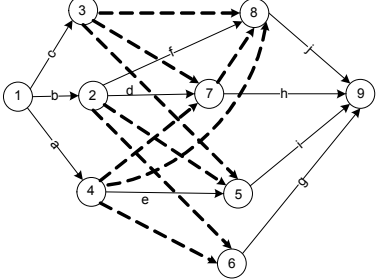
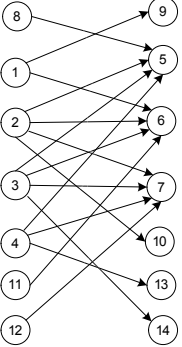
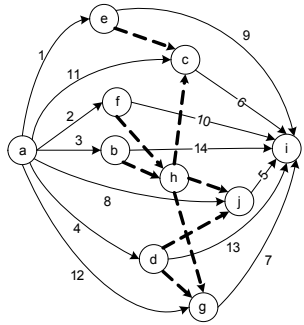
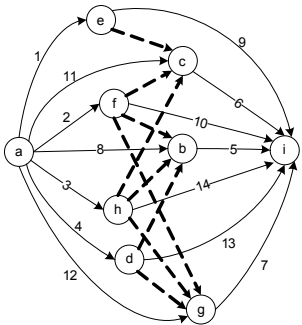
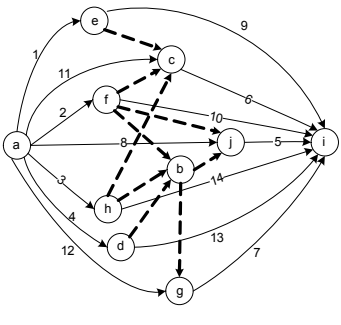
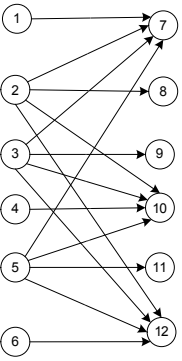
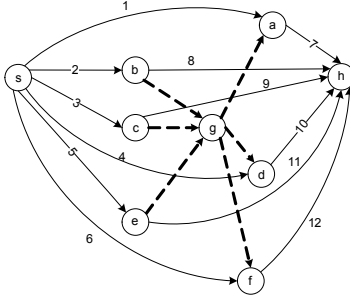
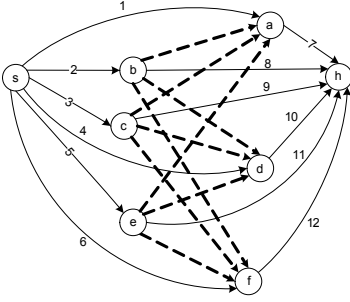
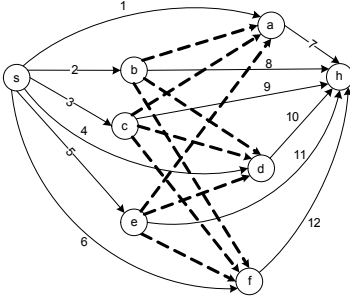
⁹ Yakın öncel ilişkilerin bozulması sorunu, Bölüm 4.1'de ayrıntılı olarak açıklanmıştır. Bir DAA'dan elde edilen KAA'da, DAA'daki yakın öncel ilişkilerin bozulmaması beklenir.

¹⁰ Önerilen yaklaşım, yakın öncel ilişkilerin bozulup bozulmayacağı ön bilgisine göre seçenekli KAA üretebilir.

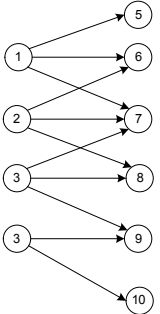
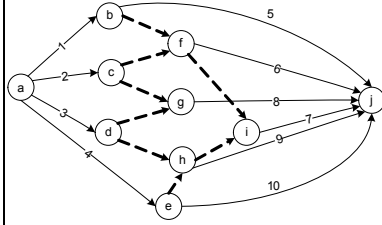
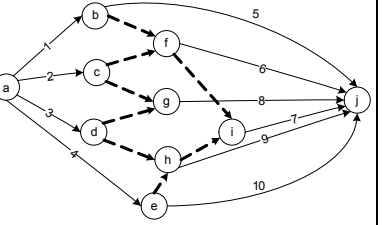
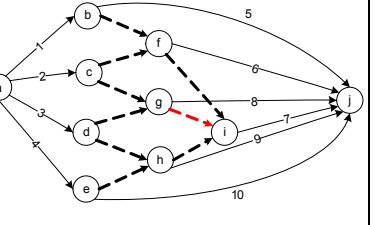
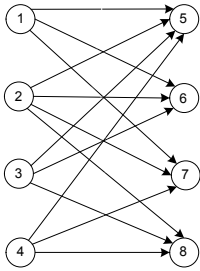
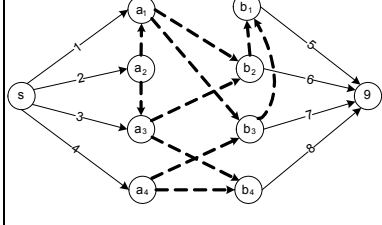
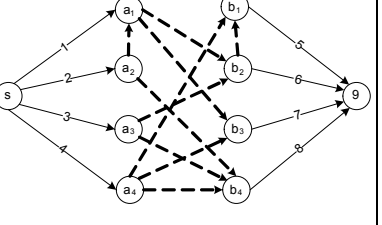
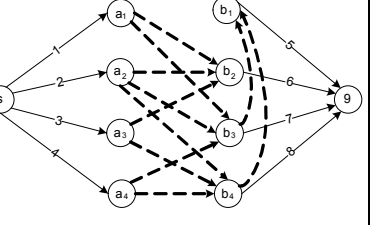
Çizelge 4.2 Literatürden alınan DAA örnekleri için, incelenen çalışmalar tarafından oluşturulan KAA'ların karşılaştırılması

Düğüm Aktif Ağ	Önerilen Yaklaşım	Michael ve diğ.	Sharma
 [5] no'lu çalışmadan alınan bir DAA	 6 düğüm, 4 yapay kenar	 6 düğüm, 4 yapay kenar	 6 düğüm, 5 yapay kenar
 [6] no'lu çalışmadan alınan bir DAA	 10 düğüm, 5 yapay kenar	 10 düğüm, 5 yapay kenar	 10 düğüm, 5 yapay kenar
 [14] no'lu çalışmadan alınan bir DAA	 6 düğüm, 4 yapay kenar	 6 düğüm, 4 yapay kenar	 6 düğüm, 4 yapay kenar
 [17] no'lu çalışmadan alınan bir DAA	 9 düğüm, 8 yapay kenar	 9 düğüm, 8 yapay kenar	 9 düğüm, 8 yapay kenar

Çizelge 4.2 Literatürden alınan DAA örnekleri için, incelenen çalışmalar tarafından oluşturulan KAA'ların karşılaştırılması(devam ediyor)

Düğüm Aktif Ağ	Önerilen Yaklaşım	Michael ve diğ.	Sharma
 [14] no'lu çalışmadan alınan bir DAA	 9 düğüm, 6 yapay kenar	 9 düğüm, 6 yapay kenar	 9 düğüm, 9 yapay kenar
 [20] no'lu çalışmadan alınan bir DAA	 10 düğüm, 8 yapay kenar	 9 düğüm, 9 yapay kenar	 10 düğüm, 9 yapay kenar
 DAA	 9 düğüm, 6 yapay kenar	 8 düğüm, 9 yapay kenar	 8 düğüm, 9 yapay kenar

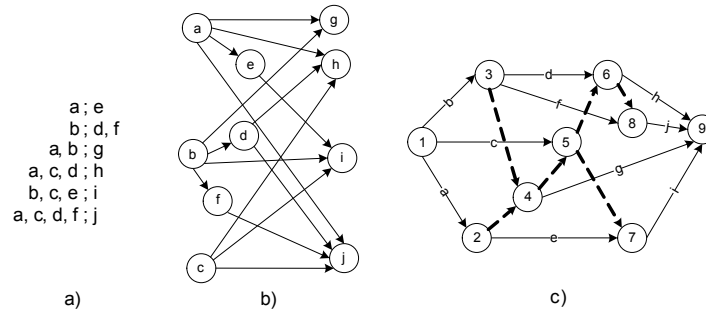
Çizelge 4.2 Literatürden alınan DAA örnekleri için, incelenen çalışmalar tarafından oluşturulan KAA'ların karşılaştırılması(devam ediyor)

Düğüm Aktif Ağ	Önerilen Yaklaşım	Michael ve diğ.	Sharma
 [11] no'lu çalışmadan alınan bir DAA	 10 düğüm, 8 yapay kenar	 10 düğüm, 8 yapay kenar	 10 düğüm, 9 yapay kenar (<g,i> fazlalık bir yapay kenardır)
 DAA	 10 düğüm, 10 yapay kenar	 10 düğüm, 10 yapay kenar	 10 düğüm, 11 yapay kenar

4.1 Yakın Öncel İlişkilerin Bozulması Sorunu

Bir KAA elde etme sürecinde, yapay kenar sayısını azaltmaya çalışırken, yakın öncel ilişkilerin bozulması istenmeyen bir durumdur. Verilen bir proje girdisi üzerinde, bu tez çalışmasında önerilen veya Michael ve diğ.[14] tarafından önerilen algoritmanın uygulanması sonucu proje girdisindeki yakın öncel ilişkilerin bozulduğu bir ağ oluşabilir.

Şekil 4.1(a)'da, [14] numaralı çalışmadan alınan bir proje girdisi verilmiştir. Bu proje girdisi üzerinde, bu tez çalışmasında önerilen veya Michael ve diğ. tarafından önerilen algoritmanın uygulanması sonucunda aynı KAA elde edilir, bu KAA, Şekil 4.1(c)'de verilmiştir. Şekil 4.1(b)'deki DAA'da, h aktivitesinin yakın öncellerinin a,c,d aktiviteleri olduğu, i aktivitesinin yakın öncellerinin b,c,e aktiviteleri olduğu, j aktivitesinin yakın öncellerinin ise a,c,d,f aktiviteleri olduğu görülmektedir. DAA'daki (b,h) aktivitesi ile ilgilenelim. b aktivitesi h aktivitesinin öncellerinden biridir, fakat yakın önceli değildir. DAA'dan elde edilen KAA'da ise b aktivitesi, h aktivitesinin yakın önceli olarak yer almaktadır. Sorun, (a,i) ve (b,j) aktivite çiftlerinde de benzerdir.

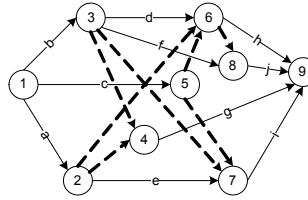


Şekil 4.1 [14] no'lu çalışmadan alınan bir proje girdisi(a), DAA(b) ve bu proje girdisi için önerilen ve Michael ve diğ.'nin yaklaşımı ile elde edilen KAA(c)

Michael ve diğ.'nin önerdiği algoritmanın, yakın öncel ilişkilerin bozulabileceği KAA'ları üretmesi olasıdır. Bu tez çalışmasında önerilen algoritma ise yapılacak bir kaç düzenleme ile söz konusu sorunu içermeyen KAA'ların üretilmesini garanti eder. Önerilen algoritmada yapılması gereken değişiklikler aşağıda verilmiştir :

- YapayKenarBüzmeKuralı1 yordamında yer alan
“if ($\forall_l \forall_k [\langle l, k \rangle \in D \vee \langle l, k \rangle \in E : l \in L, k \in K]$)” koşulu “if ($\forall_l \forall_k [\langle l, k \rangle \in D : l \in L, k \in K]$)” olarak değiştirilmelidir.
- YapayKenarEklemeKuralı3 yordamında yer alan
“if ($\forall_p \forall_s [\langle p, s \rangle \in D \vee \langle p, s \rangle \in E : p \in \mathcal{P}_x, s \in \mathcal{S}_y]$)” koşulu “f ($\forall_p \forall_s [\langle p, s \rangle \in D : p \in \mathcal{P}_x, s \in \mathcal{S}_y]$)” olarak değiştirilmelidir.

Bu tez çalışmasında önerilen algorithmada, yukarıda ifade edilen değişiklikler gerçekleştirilirse, Şekil 4.1(a)’da verilen proje girdisine karşılık Şekil 4.8’deki KAA elde edilir. Algorithmada yapılan değişiklikler sonucu, proje girdisinde verilen yakın öncelik ilişkilerinin korunduğu, fakat Şekil 4.1(b)’deki ağın içerdiğinden daha fazla yapay aktivite içeren bir ağın elde edildiği Şekil 4.2’de görülmektedir.



Şekil 4.2 Şekil 4.1(b)’de verilen DAA’daki yakın öncellerin korunduğu KAA

5. SONUÇ VE ÖNERİLER

En Az Yapay Kenar Problemi'nin NP'de *tamamlanır* problem sınıfına girmesi nedeni ile, en az sayıda yapay kenarı içeren KAA'nın bulunmasına yönelik çalışmaların izlenebilir bir algoritma ortaya koyması beklenmez. Problemin, *küme kapsama* sorununu içermesi araştırmacıları olabildiğince az yapay kenarlı KAA'ı bulmaya yönlendirmiştir.

Araştırmacıların verdikleri daha çok sayıda yapay kenar içeren KAA ile daha az yapay kenar içeren KAA arasında gerçekte bir dönüşüm yapma fırsatı olduğunu literatürde verilen örneklerden gözlemledik. Dolayısıyla, bir KAA'da öncelik ilişkilerini bozmadan, yapay kenar sayısını polinomsal zamanda azaltabilen dört sınıf indirgeme kuralı(*yapay kenar büzme*, *yapay kenar ekleme*, *düğüm örtüşme*, *düğüm ekleme*) bu çalışmada verilmiştir. İndirgeme kurallarını sınıflamadaki amaç, "hangi kuralların öncelikle uygulanması yapay kenar sayısını azaltmada daha etkili olmaktadır?" sorusunun yanıtını doğru bir mantığa dayandırmak içindir.

Bir KAA elde etme sürecinde, olabildiğince yapay kenar sayısını azaltma uğruna yakın öncellerin bozulmaması beklenir, ya da en azından problemi çözme çabası içinde olan yaklaşımların, yakın öncellerin bozulup bozulamayacağı ön bilgisine göre seçenekli KAA üretmesi daha anlamlı olur.

Önerilen yaklaşımın bilgisayar ortamında etkili gerçekleştirimi için kullanılacak veri yapıları ve indirgeme kurallarının nasıl ele alınacağı da tezde ayrıntılı olarak verilmiştir. Kenar aktif ağın oluşturulmasına ilişkin [21],[11],[14],[18]'de verilen önceki çalışmalar ayrıntılı olarak incelenmiştir. Önerilen yaklaşım, [14] ve [18]'de verilen yaklaşımlar ile algoritma karmaşıklığı, paralel kenar, geçişlilik ilişkisi, fazlalık aktivite ve yakın önceller kriterlerine bakılarak karşılaştırılmıştır. Ayrıca; önerilen ve [12],[15]'de verilen yaklaşımların literatürden alınan düğüm aktif ağlar için oluşturdukları kenar aktif ağlar, yapay aktivite ve düğüm sayıları açısından karşılaştırılmıştır.

Değişik öncelik ilişkilerini içeren düğüm aktif ağlar için, önerilen yaklaşımın ürettiği kenar aktif ağların, [14]'de verilen yaklaşımın ürettiği kenar aktif ağlar ile karşılaştırılması; önerilen yaklaşımın etkinliğini belirlemede önemli rol oynayacaktır. Değişik öncelik ilişkilerini içeren zengin bir test kümesinin üretilmesi,

arařtıřıcıların ilgisini eken bir konu olabilir. Karmařık ncellik iliřkilerinin ifade edilebildiđi DAA reteleri sz konusu olduđunda, tanımlanan indirgeme kuralların zayıf ve gl noktaları daha da netleřecektir. Kurallarda bir eksiklik varsa, arařtıřıcıların bir DAA'ı en az yapay kenarlı KAA'a dnřtren yeni bir algoritma geliřtirmek iin zaman harcamaları yerine varsa eksik polinomsal kuralın keřfedilmesine ynelik aba gstermeleri daha kayda deđer olabilir.

KAYNAKLAR DİZİNİ:

- [1] Bein W. W., Kamburowski J., Stallman M., 1992, Optimal Reduction of Two-Terminal Directed Acyclic Graphs, Volume 21, pp. 1112-1129
- [2] Boswell S.G., Eggleton R.B., 1999, An Algorithm for the Minimum Dummy Task Problem, The Journal of Combinatorial Mathematics and Combinatorial Computing, 30, pp.45-65.
- [3] Cantor D.G., Dimsdale B., 1969, On Direction-Preseving Maps of Graphs, Journal of Combinatorial Theory, 6, pp.165-176.
- [4] Corneil D.G, Gotlieb C.C, Lee Y.M, May 1973, Minimal Event-Node Network of Project Precedence Relations, Communications of the ACM, Volume 6, Number 5
- [5] De Reyck B., Herroelen W., 1994, On the Use of the Complexity Index as a Measure of Complexity in Activity Networks, European Journal of Operational Research, pp. 347-366.
- [6] Dimsdale B., March 1963, Computer Construction of Minimal Project Networks, IBM Systems Journal
- [7] Ege M. ve Ünlü A., 2004, A Fast Preprocessing Algorithm for Activity-on-Arc Construction, Teknik Rapor, Hacettepe Üniversitesi, Ankara
- [8] Ege M., Güngör K., Şen S., 2004, Kenar Aktif Ağ'da Yapay Kenarların Sayısını Azaltmak için Polinomsal İndirgeme Kuralları, Teknik Rapor, Hacettepe Üniversitesi, Ankara
- [9] Fisher A.C, Liebman J.S, Nemhauser G.R., July 1968, Computer Construction of Project Networks, Communications of the ACM, Volume II, Number 7
- [10] Gupta O. K., 1996, A Heuristic for Constructing Activity-On-Arrow Pert Networks with Minimal Dummy Activities, Operational Research Society of India, vol 33, No 2.
- [11] Güngör K., Kenar Aktif Ağ Oluşturmada Yapay Kenarların Büzülmesi, Yüksek Mühendislik Tezi, Hacettepe Üniversitesi, Fen Bilimleri Ens. Bilgisayar Bilimleri Ana Bilim Dalı, 2004
- [12] Kamburowski J., Michael D. J., Stallman M.F.M., 2000, Minimizing the Complexity of an Activity Network, Networks, vol. 36(1), pp. 47-52
- [13] Krishnamoorthy M.S, Deo N., 1979, Complexity of the Minimum-Dummy-Activities Problem in a Pert Network, Networks, Volume 9, pp. 189-194
- [14] Michael D. J., Kamburowski J. and Stallmann M.F.M., 1993, On The Minimum Dummy-Arc Problem, Operations Research, vol. 27, n^o 2, pp.153-168

- [15] Michael D. J., Kamburowski J. and Stallmann M.F.M., 1992, Optimal Construction of Project Activity Networks, Proceedings of the Decision Sciences Institute 23rd Annual Meeting, San Francisco, pp. 1424-1426
- [16] Moret B.M., 1998, The Theory of Computation, Addison-Wesley Longman, Inc., p. 453
- [17] Mrozek M., 1989, Transitively Reduced and Transitively Closed Event Networks, Networks, 1989, Volume 19, pp. 47-72
- [18] Sharma R. R. K., 1998, A New Algorithm For Preparing Pert Networks, Asia Pasific Journal of Operations Research, volume 15, pp. 37-48
- [19] Spinrad J., 1986, The Minimum Dummy Task Problem, Networks, Vol. 16, pp.331-348
- [20] Syslo M. M., 1981, Optimal Construction of Event-Node Networks, RAIRO Recherche operationelle/Operations Research, Vol. 15, no 3, pp 240-260
- [21] Ünlü A., 2002, Kenar Aktif Ağ Oluşturmak İçin Yeni Bir Yaklaşım, Yüksek Mühendislik Tezi, Hacettepe Üniversitesi, Fen Bilimleri Ens. Bilgisayar Bilimleri Ana Bilim Dalı, Ankara, 89s.

EK 1. SİMGELER VE KISALTMALAR DİZİNİ

DAA	Düğüm Aktif Ağ
KAA	Kenar Aktif Ağ
AÇVY	Ara Çizge Veri Yapısı

EK 2. SÖZLÜK

Ara çizge	Intermediary graph
Baş düğüm	Head node
Düğüm aktif ağ	Activity-on-Node(AoN) network
Düğüm ekleme	Node insertion
Düğüm örtüşme	Node overlapping
En az yapay kenar problemi	Minimum dummy arc problem
İndirgeme kuralları	Reduction rules
İzlenebilir algoritma	Tractable algorithm
Kenar aktif ağ	Activity-on-Arc(AoA) network
Kenar aktif çatı	AoA framework
Kritik yol yöntemi	Critical Path Method(CPM)
Küme kapsama problemi	Set cover problem
Kuyruk düğüm	Tail node
NP'de tamamlanır	NP-Complete
Öncelik ilişkileri	Precedence relations
Polinomsal zaman(süre)	Polynomial time
Proje değerlendirme ve gözden geçirme tekniği	Program Evaluation and Review Technique (PERT)
Sezgisel yaklaşım	Heuristic Approach
Tam ikiye ayrılır çizge	Complete bipartite graph
Yakın ardıl	Immediate successor
Yakın öncel	Immediate predecessor
Yapay kenar büzme	Dummy arc contraction
Yapay kenar ekleme	Dummy arc insertion
Yapay kenar	Dummy arc
Yönlü kenar	Directed arc