

EÜ MÜHENDİSLİK FAKÜLTESİ



EGE ÜNİVERSİTESİ

LİSANS TEZİ

**D-WAVE KUANTUM TAVLAMA SERVİSLERİYLE
YAPISAL DENGESİZLİK PROBLEMİNİNE İLİŞKİN
ÇÖZÜMLERİN FİZİBİLİTESİNİ İNCELEME**

**Ege Doğan Dursun
Cem Çorbacıoğlu**

Tez Danışmanı : Doç. Dr. Murat Osman Ünalır

Bilgisayar Mühendisliği Anabilim Dalı

Sunuş Tarihi : 12.02.2021

Bornova-İZMİR

2021

EGE ÜNİVERSİTESİ MÜHENDİSLİK FAKÜLTESİ
(LİSANS TEZİ)

**D-WAVE KUANTUM TAVLAMA SERVİSLERİYLE
YAPISAL DENGESİZLİK PROBLEMİNE İLİŞKİN
ÇÖZÜMLERİN FİZİBİLİTESİNİ İNCELEME**

**Ege Doğan DURSUN
Cem ÇORBACIOĞLU**

Tez Danışmanı: Doç. Dr. Murat Osman ÜNALIR

Bilgisayar Mühendisliği Anabilim Dalı

Sunuş Tarihi: 12.02.2021

**Bornova-İZMİR
2021**

Ege Doğan DURSUN ve Cem ÇORBACIOĞLU tarafından lisans tezi olarak sunulan “D-Wave Kuantum Tavlama Servisleriyle Yapışal Dengesizlik Problemine İlişkin Çözümlerin Fizibilitesini İnceleme” başlıklı bu çalışma Ege Üniversitesi Lisans Eğitim ve Öğretim Yönetmeliği ile Ege Üniversitesi Mühendislik Fakültesi Eğitim ve Öğretim Yönergesi'nin ilgili hükümleri uyarınca tarafımızdan değerlendirilerek savunmaya değer bulunmuş ve 12.02.2021 tarihinde yapılan tez savunma sınavında aday oybirliği/oyçokluğu ile başarılı bulunmuştur.

Jüri Üyeleri:**İmza:**

Jüri Başkanı :
Raportör Üye:

Üye :
.....

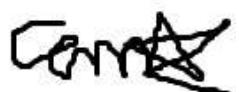
EGE ÜNİVERSİTESİ MÜHENDİSLİK FAKÜLTESİ**ETİK KURALLARA UYGUNLUK BEYANI**

Ege Üniversitesi Eğitim ve Öğretim Yönetmeliğinin ilgili hükümleri uyarınca Lisans Tezi olarak sunduğum “D-Wave Kuantum Tavlama Servisleriyle Yapısal Dengesizlik Problemine İlişkin Çözümlerin Fizibilitesini İnceleme” başlıklı bu tezin kendi çalışmam olduğunu, sunduğum tüm sonuç, doküman, bilgi ve belgeleri bizzat ve bu tez çalışması kapsamında elde ettiğimi, bu tez çalışmasıyla elde edilmeyen bütün bilgi ve yorumlara atıf yaptığımı ve bunları kaynaklar listesinde usulüne uygun olarak verdiğim, tez çalışması ve yazımı sırasında patent ve telif haklarını ihlal edici bir davranışımın olmadığını, bu tezin herhangi bir bölümünü bu üniversite veya herhangi diğer bir üniversitede başka bir tez çalışması içinde sunmadığımı, bu tezin planlanmasıdan yazımına kadar bütün sahhalarda bilimsel etik kurallarına uygun olarak davrandığımı ve aksının ortaya çıkması durumunda her türlü yasal sonucu kabul edeceğini beyan ederim.

04 / 02 / 2021



Ege Doğan DURSUN



Cem CORBACIOĞLU

ÖZET

D-WAVE KUANTUM TAVLAMA SERVİSLERİYLE YAPISAL DENGESİZLİK PROBLEMİNE İLİŞKİN ÇÖZÜMLERİN FİZİBİLİTESİNİ İNCELEME

DURSUN, Ege

ÇORBACIOĞLU, Cem

Lisans Tezi, Bilgisayar Mühendisliği Ana Bilim Dalı

Tez Danışmanı: Doç. Dr. Murat Osman ÜNALIR

Şubat 2021, 123 sayfa

Fizik yasalarının neden olduğu engellere bağlı olarak, klasik bilgisayarların gelişimi son yıllarda beklenenden yavaş ilerlemektedir. İllerleyen teknoloji sayesinde transistörler, zamanla küçülmeye devam etmek ile birlikte ulaşabilecekleri minimum ebat bir atomdan daha küçük olamaz. Moore yasasının kaçınılmaz sonunu getirecek bu durum ile ilgili günümüzde yükseliş geçen önemli teknolojilerden biri kuantum bilgisayarlarıdır. Kuantum bilgisayarlar, klasik fizik kurallarından bağımsız olarak kuantum fizik yasalarına göre işlemekte ve fonksiyonlarını bu alana has dolanıklık, süperpozisyon, belirsizlik ve interferans ilkeleri sayesinde gerçekleştirmektedir. Günümüz klasik bilgisayarlarının sahip olduğu mantık kapılarının kuantum eşleniğine sahip olması amaçlanan dijital kuantum bilgisayarlar emekleme aşamasında olmasına rağmen; analog çalışma prensibine sahip Kuantum Tavlama Sistemlerinin deneysel ve endüstriyel kullanımlarına başlanmıştır. Çalışmamız dahilinde, D-Wave'in sunduğu Ocean SDK kütüphanesini kullanarak, Kuantum Tavlama bulut servislerinin başta asayış, iç/dış güvenlik, organizasyonel ve toplumsal düzen olmak üzere birçok konuda çözümler getirebilecek Yapısal Dengesizlik problemi olmak üzere çeşitli yüneylem problemleri konusunda ulaştığı noktayı sınamayı ve klasik işlemci performansları ile kıyaslamayı amaçlıyoruz.

Anahtar Sözcükler: Analog Kuantum Bilgisayar, İşaretli Sosyal Çizgeler, D-Wave Kuantum Tavlama Bulut Servisleri, Yöneylem Araştırması, Yapısal Dengesizlik Problemi

ABSTRACT

INSPECTING THE FEASIBILITY OF THE SOLUTIONS TO STRUCTURAL IMBALANCE PROBLEM SOLVED BY D- WAVE QUANTUM ANNEALING SERVICES

DURSUN, Ege

ÇORBACIOĞLU, Cem

Undergraduate Thesis, Computer Engineering Department

Supervisor: Assoc. Prof. Dr. Murat Osman ÜNALIR

February 2021, 123 pages

Due to obstacles caused by the laws of physics, the development of classical computers has been slower than expected in recent years. Although transistors continue to shrink thanks to advancing technology, the minimum size they can reach cannot be smaller than an atom. Quantum computers are one of the important technologies that are on the rise today regarding this situation that will inevitably end Moore's law. Quantum computers operate according to the laws of quantum physics independently from laws of classical physics and perform their functions thanks to the principles of entanglement, superposition, uncertainty and interference which are specific to this area. Although digital quantum computers, which are intended to have the quantum equivalent of the logic gates of today's classical computers, are in their infancy; the experimental and industrial uses of Quantum Annealing Systems with analog working principle have been started. In our study, using the Ocean SDK library offered by D-Wave, Quantum Annealing In our study, by using D-Wave's Ocean SDK, we aim to test the feasibility of Quantum Annealing Services and compare their performance with classical processors for various real world operational research problems and primarily Structural Imbalance problem, which could provide advantage in law enforcement, internal/external security, organisational and social order.

Keywords: Analog Quantum Computer, Signed Social Graphs, D-Wave Quantum Annealing Cloud Services, Operational Research Problems, Structural Imbalance Problem

TEŞEKKÜR

Bu çalışma dahilinde Kuantum Tavlama servislerini ve Kuantum İşlemci Birimlerini bulut üzerinden ücretsiz olarak kullanmamıza olanak sunan D-Wave'e ve kıymetli görüşlerinden faydalandığımız tez danışmanımız sayın Doç. Dr. Murat Osman Ünalır'a teşekkürü bir borç biliriz.

İÇİNDEKİLER

	<u>Sayfa</u>
ÖZET	vii
ABSTRACT	ix
TEŞEKKÜR	xi
ŞEKİLLER DİZİNİ.....	xvi
1.GİRİŞ	1
2.LİTERATÜR ÇALIŞMASI	12
3.YÖNTEM VE TEKNOLOJİLER	14
4.ANALİZ VE PROBLEM TASARIMI	15
5.NETWORK-X KÜTÜPHANESİ İLE ÇİZGE OLUŞTURMA	17
6.D-WAVE QPU VE CPU İLE ÖRNEK PROBLEMLERİN ÇÖZÜMÜ.....	29
6.1 Minimum Vertex Cover Problemi	30
6.2 Map Coloring Problemi	39
6.3 Maximum Clique Problemi	46
6.4 Minimum Maximal Matching Problemi	52
6.5 Maximum Cut Problemi	58
6.6 Traveling Salesperson Problemi	64
6.7 Structural Imbalance Problemi	70

İÇİNDEKİLER (devam)

Sayfa

7.D-WAVE QPU/CPU HİBRİTLEME VE CPU İLE ÖRNEK ÇÖZÜMLER ...	79
7.1 Minimum Vertex Cover Problemi	80
7.2 Map Coloring Problemi	83
7.3 Maximum Clique Problemi	86
7.4 Minimum Maximal Matching Problemi	89
7.5 Maximum Cut Problemi	92
7.6 Traveling Salesperson Problemi	95
7.7 Structural Imbalance Problemi	98
8.YAPISAL DENGESİZLİK PROBLEMLERİNİN ÇÖZÜMLENMESİ.....	103
8.1 Örnek Yapısal Dengesizlik Problemlerinin Tanımlanması.....	104
8.2 Problemlerin CPU Kullanılarak Çözümlenebilirliği	108
8.3 Problemlerin D-Wave QPU Kullanılarak Çözümlenmesi	109
8.4 Problemlerin D-Wave QPU/CPU Hibritleme Kullanılarak Çözümlenmesi .	110
8.5 Sonuçların Karşılaştırılması	112
9.TARTIŞMA	116
10.SONUÇ	117

İÇİNDEKİLER (devam)Sayfa

KAYNAKLAR DİZİNİ.....	119
ÖZGEÇMİŞ	122
EKLER	123

ŞEKİLLER DİZİNİ

<u>Sekil</u>	<u>Sayfa</u>
1.1. Bloch Küresi	3
1.2. Dijital Kuantum Bilgisayarı	5
1.3. D-Wave Analog Kuantum Bilgisayarı	6
1.4. İşaretli Sosyal Çizge	8
1.5. Üç birimden oluşan sosyal çizge örnekleri	8
1.6. Büyük-O Zaman Karmaşıklığı grafiği	10
5.1. Network-X. Örnek Complete Graph 1	18
5.2. Network-X. Örnek Complete Graph 2	18
5.3. Network-X. Örnek Balanced Tree 1	19
5.4. Network-X. Örnek Balanced Tree 2	19
5.5. Network-X. Örnek Barbell Graph 1	20
5.6. Network-X. Örnek Barbell Graph 2	20
5.7. Network-X. Örnek Circular Ladder Graph 1	21
5.8. Network-X. Örnek Circular Ladder Graph 2	21
5.9. Network-X. Örnek Cycle Graph 1	22
5.10. Network-X. Örnek Cycle Graph 2	22
5.11. Network-X. Örnek Ladder Graph 1	23

ŞEKİLLER DİZİNİ (devam)

<u>Sekil</u>	<u>Sayfa</u>
5.12. Network-X. Örnek Ladder Graph 2	23
5.13. Network-X. Örnek Lollipop Graph 1	24
5.14. Network-X. Örnek Lollipop Graph 2	24
5.15. Network-X. Örnek Star Graph 1	25
5.16. Network-X. Örnek Star Graph 2	25
5.17. Network-X. Örnek Turan Graph 1	26
5.18. Network-X. Örnek Turan Graph 2	26
5.19. Network-X. Örnek Wheel Graph 1	27
5.20. Network-X. Örnek Wheel Graph 2	27
5.21. Network-X. Örnek Özel Ağırlıklı Çizge	28
6.1. Gerekli kütüphanelerin import edilmesi	29
6.1.1. Minimum Vertex Cover, 1. Problem	30
6.1.2. Minimum Vertex Cover, 2. Problem	31
6.1.3. Minimum Vertex Cover, 3. Problem	32
6.1.4. Minimum Vertex Cover, 1. Problemin QPU ile Çözümlenmesi.....	32
6.1.5. Minimum Vertex Cover, 1. Problemin CPU ile Çözümlenmesi	33
6.1.6. Minimum Vertex Cover, 2. Problemin QPU ille Çözümlenmesi	33
6.1.7. Minimum Vertex Cover, 2. Problemin CPU ile Çözümlenmesi	34

ŞEKİLLER DİZİNİ (devam)

<u>Sekil</u>	<u>Sayfa</u>
6.1.8. Minimum Vertex Cover, 3. Problemin QPU ile Çözümlenmesi	34
6.1.9. Minimum Vertex Cover, 3. Problemin CPU ile Çözümlenmesi	35
6.1.10. Minimum Vertex Cover, Çözümlerin Karşılaştırılması	36
6.1.11. QPU ve CPU Ortalama İşlem Gecikmesi Tablosu	37
6.1.12. QPU ve CPU İşlem Gecikmesi Grafiği	38
6.2.1. Map Coloring, 1. Problem	39
6.2.2. Map Coloring, 2. Problem	40
6.2.3. Map Coloring, 3. Problem	41
6.2.4. Map Coloring, 1. Problemin QPU ile Çözümlenmesi	41
6.2.5. Map Coloring, 1. Problemin CPU ile Çözümlenmesi	42
6.2.6. Map Coloring, 2. Problemin QPU ile Çözümlenmesi	42
6.2.7. Map Coloring, 2. Problemin CPU ile Çözümlenmesi	43
6.2.8. Map Coloring, 3. Problemin QPU ile Çözümlenmesi	43
6.2.9. Map Coloring, 3. Problemin CPU ile Çözümlenmesi	44
6.2.10. Map Coloring, QPU ve CPU Çözümlerinin Karşılaştırılması	44
6.3.1. Maximum Clique, 1. Problem	46
6.3.2. Maximum Clique, 2. Problem	47
6.3.3. Maximum Clique, 3. Problem	48
6.3.4. Maximum Clique, 1. Problemin QPU ile Çözümlenmesi	48

ŞEKİLLER DİZİNİ (devam)

<u>Sekil</u>	<u>Sayfa</u>
6.3.5. Maximum Clique, 1. Problemin CPU ile Çözümlenmesi	49
6.3.6. Maximum Clique, 2. Problemin QPU ile Çözümlenmesi	49
6.3.7. Maximum Clique, 2. Problemin CPU ile Çözümlenmesi	50
6.3.8. Maximum Clique, 3. Problemin QPU ile Çözümlenmesi	50
6.3.9. Maximum Clique, 3. Problemin CPU ile Çözümlenmesi	51
6.3.10. Maximum Clique, QPU ve CPU Çözümlerinin Karşılaştırılması	51
6.4.1. Minimum Maximal Matching, 1. Problem	52
6.4.2. Minimum Maximal Matching, 2. Problem	53
6.4.3. Minimum Maximal Matching, 3. Problem	54
6.4.4. Minimum Maximal Matching, 1. Problemin QPU ile Çözümlenmesi	54
6.4.5. Minimum Maximal Matching, 1. Problemin CPU ile Çözümlenmesi	55
6.4.6. Minimum Maximal Matching, 2. Problemin QPU ile Çözümlenmesi	55
6.4.7. Minimum Maximal Matching, 2. Problemin CPU ile Çözümlenmesi	56
6.4.8. Minimum Maximal Matching, 3. Problemin QPU ile Çözümlenmesi	56
6.4.9. Minimum Maximal Matching, 3. Problemin CPU ile Çözümlenmesi	57
6.4.10. QPU ve CPU Ölçümlerinin Karşılaştırılması	57
6.5.1. Maximum Cut, 1. Problem	58
6.5.2. Maximum Cut, 2. Problem	59
6.5.3. Maximum Cut, 3. Problem	60

ŞEKİLLER DİZİNİ (devam)

<u>Sekil</u>	<u>Sayfa</u>
6.5.4. Maximum Cut, 1. Problemin QPU ile Çözümlenmesi	60
6.5.5. Maximum Cut, 1. Problemin CPU ile Çözümlenmesi	61
6.5.6. Maximum Cut, 2. Problemin QPU ile Çözümlenmesi	61
6.5.7. Maximum Cut, 2. Problemin CPU ile Çözümlenmesi	62
6.5.8. Maximum Cut, 3. Problemin QPU ile Çözümlenmesi	62
6.5.9. Maximum Cut, 3. Problemin CPU ile Çözümlenmesi	63
6.5.10. Maximum Cut, QPU ve CPU Çözümlerinin Karşılaştırılması	63
6.6.1. Traveling Salesperson, 1. Problem	64
6.6.2. Traveling Salesperson, 2. Problem	65
6.6.3. Traveling Salesperson, 3. Problem	66
6.6.4. Traveling Salesperson, 1. Problemin QPU ile Çözümlenmesi	66
6.6.5. Traveling Salesperson, 1. Problemin CPU ile Çözümlenmesi	67
6.6.6. Traveling Salesperson, 2. Problemin QPU ile Çözümlenmesi	67
6.6.7. Traveling Salesperson, 2. Problemin CPU ile Çözümlenmesi	68
6.6.8. Traveling Salesperson, 3. Problemin QPU ile Çözümlenmesi	68
6.6.9. Traveling Salesperson, 3. Problemin CPU ile Çözümlenmesi	69
6.6.10. Traveling Salesperson, QPU ve CPU Çözümlerinin Karşılaştırılması	69

ŞEKİLLER DİZİNİ (devam)

<u>Sekil</u>	<u>Sayfa</u>
6.7.1. Structural Imbalance, 1. Problem	70
6.7.2. Structural Imbalance, 1. Problem Çizge	71
6.7.3. Structural Imbalance, 2. Problem	71
6.7.4. Structural Imbalance, 2. Problem Çizge	72
6.7.5. Structural Imbalance, 3. Problem	72
6.7.6. Structural Imbalance, 3. Problem Çizge	73
6.7.7. Structural Imbalance, 1. Problemin QPU ile Çözümlenmesi.....	73
6.7.8. Structural Imbalance, 1. Problemin CPU ile Çözümlenmesi	73
6.7.9. Structural Imbalance, 2. Problemin QPU ile Çözümlenmesi	74
6.7.10.Structural Imbalance, 2. Problemin CPU ile Çözümlenmesi	74
6.7.11.Structural Imbalance, 3. Problemin QPU ile Çözümlenmesi	75
6.7.12.Structural Imbalance, 3. Problemin CPU ile Çözümlenmesi	75
6.7.13.Structural Imbalance, 1. Problem, QPU Sonuç	75
6.7.14.Structural Imbalance, 1. Problem, CPU Sonuç	76
6.7.15.Structural Imbalance, 2. Problem, QPU Sonuç	76
6.7.16.Structural Imbalance, 2. Problem, CPU Sonuç	77
6.7.17.Structural Imbalance, 3. Problem, QPU Sonuç	78
6.7.18.Structural Imbalance, 3. Problem, CPU Sonuç	78
7.1. D-Wave QPU/CPU Hibrit Çözümleyicinin Import Edilmesi	79

ŞEKİLLER DİZİNİ (devam)

<u>Sekil</u>	<u>Sayfa</u>
7.1.1. Minimum Vertex Cover, Problem	80
7.1.2. Minimum Vertex Cover, Problemin QPU/CPU Hibrit ile Çözümlenmesi .	80
7.1.3. Minimum Vertex Cover, Problemin QPU ile Çözümlenmesi	81
7.1.4. Minimum Vertex Cover, Problemin CPU ile Çözümlenmesi	81
7.1.5. Minimum Vertex Cover, Çözümlerin Karşılaştırılması	82
7.2.1. Map Coloring, Problem	83
7.2.2. Map Coloring, Problemin QPU/CPU Hibrit ile Çözümlenmesi	84
7.2.3. Map Coloring, Problemin QPU ile Çözümlenmesi	84
7.2.4. Map Coloring, Problemin CPU ile Çözümlenmesi	85
7.2.5. Map Coloring, Çözümlerin Karşılaştırılması	85
7.3.1. Maximum Clique, Problem	86
7.3.2. Maximum Clique, Problemin QPU/CPU Hibrit ile Çözümlenmesi	87
7.3.3. Maximum Clique, Problemin QPU ile Çözümlenmesi	87
7.3.4. Maximum Clique, Problemin CPU ile Çözümlenmesi	87
7.3.5. Maximum Clique, Çözümlerin Karşılaştırılması	88
7.4.1. Minimum Maximal Matching, Problem	89
7.4.2. Minimum Maximal Matching, Problemin QPU/CPU ile Çözümlenmesi ..	89
7.4.3. Minimum Maximal Matching, Problemin QPU ile Çözümlenmesi	90
7.4.4. Minimum Maximal Matching, Problemin CPU ile Çözümlenmesi	90

ŞEKİLLER DİZİNİ (devam)

<u>Sekil</u>	<u>Sayfa</u>
7.4.5. Minimum Maximal Matching, Çözümlerin Karşılaştırılması	91
7.5.1. Maximum Cut, Problem	92
7.5.2. Maximum Cut, Problemin QPU/CPU Hibrit ile Çözümlenmesi	92
7.5.3. Maximum Cut, Problemin QPU ile Çözümlenmesi	93
7.5.4. Maximum Cut, Problemin CPU ile Çözümlenmesi	93
7.5.5. Maximum Cut, Çözümlerin Karşılaştırılması	94
7.6.1. Traveling Salesperson, Problem	95
7.6.2. Traveling Salesperson, Problemin QPU/CPU hibrit ile Çözümlenmesi ...	95
7.6.3. Traveling Salesperson, Problemin QPU ile Çözümlenmesi	96
7.6.4. Traveling Salesperson, Problemin CPU ile Çözümlenmesi	96
7.6.5. Traveling Salesperson, Çözümlerin Karşılaştırılması	97
7.7.1. Structural Imbalance, Problem	98
7.7.2. Structural Imbalance, Problem Çizge.....	99
7.7.3. Structural Imbalance, Problemin QPU/CPU hibrit ile Çözümlenmesi	99
7.7.4. Structural Imbalance, Problemin QPU ile Çözümlenmesi	99
7.7.5. Structural Imbalance, Problemin CPU ile Çözümlenmesi	100
7.7.6. Structural Imbalance, QPU/CPU Çözümleri	100
7.7.7. Structural Imbalance, QPU Çözümleri	101

ŞEKİLLER DİZİNİ (devam)

<u>Sekil</u>	<u>Sayfa</u>
7.7.8. Structural Imbalance, CPU Çözümleri	101
8.1.1. Yapısal Dengesizlik Problemi, 1. Problem	104
8.1.2. Yapısal Dengesizlik Problemi, 1. Problem Çizge	105
8.1.3. Yapısal Dengesizlik Problemi, 2. Problem	105
8.1.4. Yapısal Dengesizlik Problemi, 2. Problem Çizge	106
8.1.5. Yapısal Dengesizlik Problemi, 3. Problem	106
8.1.6. Yapısal Dengesizlik Problemi, 3. Problem Çizge	107
8.3.1. Yapısal Dengesizlik Problemi, 1. Problemin QPU ile Çözümlenmesi	109
8.3.2. Yapısal Dengesizlik Problemi, 2. Problemin QPU ile Çözümlenmesi.....	109
8.3.3. Yapısal Dengesizlik Problemi, 3. Problemin QPU ile Çözümlenmesi	109
8.4.1. Yapısal Dengesizlik Problemi, 1.Problemin QPU/CPU Çözümlenmesi ..	110
8.4.2. Yapısal Dengesizlik Problemi, 2.Problemin QPU/CPU Çözümlenmesi ..	110
8.4.3. Yapısal Dengesizlik Problemi, 3.Problemin QPU/CPU Çözümlenmesi ..	111
8.5.1. Yapısal Dengesizlik Problemi, 1. Problem QPU Çözümleri	112
8.5.2. Yapısal Dengesizlik Problemi, 1. Problem QPU/CPU Çözümleri	112
8.5.3. Yapısal Dengesizlik Problemi, 2. Problem QPU Çözümleri	113
8.5.4. Yapısal Dengesizlik Problemi, 2. Problem QPU/CPU Çözümleri	114
8.5.5. Yapısal Dengesizlik Problemi, 3. Problem QPU Çözümleri	114

ŞEKİLLER DİZİNİ (devam)SekilSayfa

8.5.6. Yapısal Dengesizlik Problemi, 3. Problem QPU/CPU Çözümleri 115

1. GİRİŞ

1940'lardan itibaren geçen her on yılda bilgisayarların hayatımızdaki yeri git gide artmış ve yükselen işlem güçleri ile birlikte kullanım kapsamları da genişlemiştir. Yalnızca performans açısından değil; kapladıkları alan konusunda da önemli gelişmeler yaşamış olan bilgisayarlar, üretildikleri ilk dönemlerde birkaç oda büyülüğündeyken günümüzde o döneme kıyasla binlerce kat daha etkin bir çalışma performansını yalnızca bir kibrit kutusu büyülüğündeki cihazlar ile sağlayabilmektedirler.

Bilgisayarların boyutunun küçülmesinin en büyük sebebi transistör teknolojisinde yaşanan gelişmelerdir. Transistörlerin boyutunun zaman içerisinde küçülmesi, birim alana yerleştirilecek transistör sayısını artırması ile birlikte bilgisayarların birim zamanda gerçekleştirebilecekleri işlem performansını da olumlu yönde etkilemiştir. Transistörlerin zaman içerisindeki küçülme hızlarını keşfeden Gordon E. Moore, 1965 yılında Moore yasasını.^[1] ortaya koymuştur. Moore yasasına göre entegre devre başına düşen transistör sayısı her yıl 2 katına çıkmaktadır. 1965 yılında ortaya atılan bu tahminin gelecek 10 yıl boyunca geçerliliğini koruyacağını düşünen Moore, 1975 yılında Moore yasasını, entegre devrelere yerleştirilebilen transistör sayısının her yıl yerine, her 2 yılda bir 2 katına çıkacağı yönünde güncellemiştir^[2]. Moore her ne kadar öne sürdüğü tahminle ilgili deneysel bir çalışma ya da kanıt ortaya koymamış olsa da bu tahmin endüstride 50 yıldan uzun bir süre geçerliliğini korumaya devam etmiş ve mevcut işlemcilerin performansı üzerinde yapılan çalışmalar Moore yasasının yaklaşık geçerliliğini ortaya koymuştur.^{[3][4][5]} Moore yasasının ne zamana kadar geçerliliğini südüreceği konusunda akademik veya sektörel çevrelerce henüz bir uzlaşmaya varılabilmiş değildir ancak gözlemler 2010'lu yıllarda itibaren transistör teknolojisinde yaşanan gelişmelerin eskisi kadar hızlı olmadığını ortaya koymaktadır. ^[6]

Mevcut teknolojilik gelişmelerin yavaşlığı göz önünde bulundurulursa Moore yasasının uzun vadede işlerliğini yitirmesi kaçınılmazdır. Bunun en büyük sebeplerinden biri transistörlerin boyutunun küçülmesinin yarattığı beklenmedik yan etkilerdir. Transistörlerin boyutu küçüldükçe, klasik fizik yasalarının etkisi azalmakta ve nesnelere ne kadar küçük boyuttaysa etkileri o kadar aktif boyutta hissedilen kuantum fizik yasalarının etkisi artmaktadır. ^[7] Buna ek olarak temel

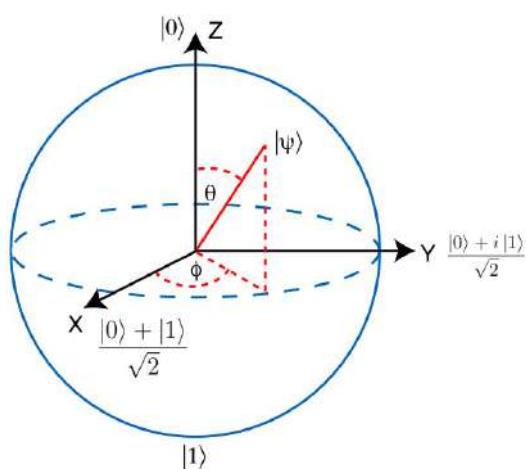
işlevi elektron akışını kontrol etmek olan transistörlerin boyutunun bir atomdan daha küçük olması onları kullanışız hale getireceğinden ulaşabilecekleri minimum boyut limitlenmektedir. Günümüzde üretilmiş en küçük transistörün boyutu 2.5 nanometredir ^[8] ve bu boyut bir atomun yaklaşık çapının 5 ile 25 katı arasındadır. ^[9] Moore yasasının benzer hızlarda ilerleyecek olması durumunda öümüzdeki 10-20 yıllık sürecin ardından transistör teknolojisinin ilerleyebileceği nokta önceden tahminlenebilir ve kaçınılmaz bir çıkmaza girecektir. Bu sorun, birçok akademik ve endüstriyel çevre tarafından yıllar önce fark edilmiş ve çözümüne yönelik arayışlara girilmiştir. İşlemcilerin frekanslarını artırma konusunda da belli noktadan sonra ısınma konusunda sıkıntılar yaşayan entegre devre şirketleri öncelikli olarak işlemci çekirdeklerinin sayısını artırma yoluna girmiştir.

Transistörler konusunda üretilen çözümlerden bir tanesi ise silikon kadar iletkenlik özelliği gösteren ancak ısiya silikondan çok daha dayanıklı, tek atom kalınlığındayken bile stabil olarak işlev görebilen bir madde olan grafendir. ^[10] Grafen teknolojisinin, Moore yasasının kaçınılmaz kıyametini, daha etkin ısi kontrolü ile birlikte işlemci frekanslarında yaşanabilecek artışlara bağlı olarak bir miktar daha ertelemesi beklenmektedir.

Klasik bilgisayarların gelişimini kurtarmaktansa daha devrimsel ve yenilikçi bir yaklaşım gerçekleştirilmesi gerektiği fikri ise ilk defa Paul Benioff'un 1980'lerin başlarında Turing makinesinin kuantum mekanik eşleniğini tasarlamasının ardından kuantum bilgisayar kavramı ile ortaya çıkmıştır. ^[11] Sonrasında bu konu ile ilgili görüşlerini belirtmiş olan Richard Feynman'a göre ^[12] kuantum bilgisayarlar klasik bilgisayarların çözüm getiremediği belirli problemlere çözüm getirmek konusunda oldukça avantajlı olabilecekti. Buna ek olarak, transistörlerin gittikçe küçülmesinin etki artışına yol açtığı kuantum olaylar, klasik bir bilgisayardansa kuantum bilgisayarlar kullanılarak fayda doğrultusunda kullanılması yeni teknolojik gelişmelere zemin hazırlayabilecek ve çözülmesi güç olan problemlerin çözümünde avantaj sağlayabilecekti. 1994 yılında Peter Shor, tamsayı faktörizasyonu kullanılarak RSA ile şifrelenmiş iletişimleri deşifre etme potansiyeli taşıyan bir kuantum algoritma geliştirmiştir. ^[13] 1990'lı yillardan beri kuantum bilgisayarlar ile ilgili ciddi gelişmeler yaşanmasına rağmen; hata toleransı yüksek kuantum bilgisayarları belirli akademik çevrelere göre hala uzak bir hayaldır. Hata toleransının yüksek olması kuantum bilgisayarların potansiyeline yönelik en önemli ölçütlerden biridir çünkü bu faktör, kuantum bilgisayarların klasik bilgisayarlara kıyasla özellikle NP-Complete ve NP-Hard problem

karmaşıklıklarına sahip işlemlerde ne kadar daha etkin çalışabileceğini belirlemektedir.

Kuantum bilgisayarların çalışma yöntemi klasik bilgisayarlara göre belirli açılardan farklılık göstermektedir. Klasik bilgisayarlarda ifade edilen veriler ikili sayı sistemindeki bitler ile ifade edilirken, kuantum bilgisayarlarda ifade edilen veriler qubitler ile ifade edilmektedir. Qubit'lerin bitlere göre en temel farkı bitler gibi 1 veya 0 değerlerini alabilmelerine ek olarak; bu değerin her ikisini de süperpozisyon dahilinde aynı anda taşıyabilmeleridir. Qubitlerin süperpozisyondaki bu durumu ancak ve ancak ölçümlenmedikleri anlarda geçerlidir. Ölçümlenmemiş Qubit'ler, süperpozisyon durumunda iken 1 veya 0 olmaları yönündeki olasılıklar şeklinde ifade edilirken, ölçümlendikleri durumda dalga fonksiyonunda yaşanan çökme sonucunda bu olasılıksal değerlere bağlı olarak ya 1, ya da 0 ifadelerine dönüşeceklərdir. Bu durum kuantum fiziği dahilinde Schrödinger'in kedisi örneği ile ifade edilmektedir. [\[14\]](#) Bu örneğe göre, dışarıdan hiçbir duyu organı ile gözlemlenemeyen bir kutu içerisinde duran bir kedi vardır. Aynı kutunun içerisinde de bozunma ihtimali tam olarak %50 olan bir radyoaktif izotop bulunmaktadır. Eğer radyoaktif bozunma olursa, kedi ölüür. Olmaz ise kedi yaşar. Kutunun dışarısından içinde ne olduğu asla gözlemlenemediği bilindiğine göre, kuantum yasaları gereği kedi aynı anda hem canlı, hem de ölüdür. Fakat kutu açılıp kedinin durumu gözlemlendiği anda dalga fonkiyonu çöküşe uğrar ve sonuç olarak kedi ya canlı, ya da ölü olarak görülür.



Şekil 1.1. Bloch Küresi

1 ve 0 dışında alınabilecek süperpozisyonaya dayalı olasılıksal değerlerin var olması, Qubit’ler ifade edilirken Bloch Küresinin kullanımını avantajlı kılmıştır. Klasik bitler ve Qubit’ler veri depolama özellikleri açısından da farklılık göstermektedirler. ([Şekil 1.1](#)) Örneğin 2 klasik bit, 2 bitlik veri depolayabilmektedir. Oysa 2 bitlik Qubit, sahip oldukları değeri ölçümleyebilmek için 4 değişken gerektirdiğinden 4 bitlik veri ifade etmektedir. Buna dayalı olarak ideal koşullarda N adet Qubit'in ifade ettiği klasik bit sayısı 2^n olmaktadır. Fakat, günümüz kuantum bilgisayarlarında bu kadar ideal bir senaryoya erişimini engelleyen birkaç faktör vardır. Bunlara gürültü, bozulma süresi, yetersiz hata toleransı ve yüksek sayıda Qubit'i birbirine dolanıklaştımanın zorluğunu örnek verebiliriz. Verileri ifade etmekte yukarıdaki formülde belirtildiği kadar ideal bir performans artışı sağlanabilmesi için Qubit'lerin hepsinin birbirine ideal olarak dolanıklaştırılabilmesi şarttır. Dolanıklık, kuantum fiziğindeki bir başka kavramdır ve bir parçacığın arasındaki mesafe fark etmemesinin bir başka parçacığa etkiyebilmesini ifade etmektedir. Kuantum yasaları gereği evrenin herhangi bir noktasında bulunan bir elektronun spininde yaşanan bir değişiklik, dolanık olduğu çifti evrenin hangi köşesinde olursa olsun onda da gözlemlenecektir. Bu ilginç durum, Kuantum bilgisayarların klasik yöntemlerle çözümlenmesi güç olan NP-Complete ve NP-Hard problemlere çözüm getirmeye ek olarak hızlı veri transferi veya veri işinlama gibi konularda da kullanılabilirnesine yönelik meraka yol açmıştır. Klasik bilgisayarların çözmekte zorlandığı ve kuantum bilgisayarlarının çözüm getirebilceği problemlere örnek olarak Map Coloring problemi, Traveling Salesperson problemi, Minimum Vertex Cover problemi ve Sosyal Çizge problemi gösterilebilir. Pratikte bu problemleri üssel zamanda çözümleyebilen klasik bilgisayarlara karşın, kuantum bilgisayarları teorik olarak bu problemlere lineer zamanda çözüm getirebilecektir. Buna ek olarak, süper pozisyonun yarattığı belirsizlik kuantum fiziği dahilinde şifrelenmiş verilerin gözlemlendikleri durumda gerçek doğalarını tamamen kaybetmeleri söz konusu olduğundan kuantum bilgisayarlarını kriptografi açısından eşsiz bir aday kılmaktadır. Pratikte, kuantum kriptografi ile şifrelenmiş bir verinin şifresinin kırılması, ancak ve ancak kuantum fiziği kanunlarını aşarak mümkün olabilmektedir.

Kuantum bilgisayarlarının klasik bilgisayarlardan daha etkin olarak çözüm getirebileceği konulara verilebilecek örnekler aşağıdaki gibidir:

1. Parçacık Fiziği Simülasyonları
2. Kimyasal Etkileşim Simülasyonları
3. NP-Complete ve NP-Hard Zaman Karmaşıklığına Sahip Problemlerin Çözümlenmesi
4. Tıbbi İlaç Formülizasyonları
5. Kuantum Makine Öğrenmesi
6. Siber Güvenlik ve Kriptografi
7. Finansal Modellemeler ve Tahminlemeler

Dijital kuantum bilgisayarların oluşturulabilmesi için teknoloji geliştirme çabaları hala devam etmektedir. Ancak 2020 yılı itibarı ile gelinen nokta hala dijital kuantum bilgisayarlarının endüstriyel kullanımına izin verecek düzeyde değildir. Başta Avrupa Birliği, Amerika Birleşik Devletleri, Rusya ve Çin dahilinde kuantum bilgisayarların geliştirilmesine yönelik çabalar aktif olarak yürütülmekte ve desteklenmektedir. Dijital kuantum bilgisayarlarının geliştirilmesine karşı çaba yürüten şirketlere başta IBM, Google, Microsoft gibi sektör devlerinin yanında Finlandiya ve Almanya'da çalışmalarını yürüten IQM gösterilebilir. [\[15\]](#) [\[16\]](#) [\[17\]](#) [\[18\]](#)



Şekil 1.2. Dijital Kuantum Bilgisayarı

Dijital kuantum bilgisayarlarının oluşturulmasında engel oluşturan temel problemlere örnek olarak kuantum mantık kapılarının oluşturulmasının zorluğu, Qubit'lerin bozunma sürelerinin çok kısa oluşu gösterilebilir. Çeşitli fizibilite problemlerinden ötürü dijital kuantum bilgisayarlarını oluşturmak için kullanılan teknoloji ve materyaller farklılık göstermektedir. Bu teknolojilere verilebilecek iki örnek iyon tuzakları ve süper iletkenlerdir. Bu teknolojilerden en çok gelecek vaat ettiğine inanılan ise süper iletkenlerdir. Süper iletkenler kritik sıcaklığın altında sıfır direnç özelliğini gösteren materyallerdir. Dijital kuantum bilgisayarlar söz konusu olduğunda önceden bahsedilen kuantum fiziksel özelliklerin sağlanabilmesi ve Qubit'lerin bozunma sürelerinin uzatılması gibi sebeplerden dolayı süper iletkenler avantajlı konumdadır. Bu tür kuantum bilgisayarlarının işlem birimleri kriyostat adı verilen özel soğutma ünitelerinin içinde yer almaktadır. ([Şekil 1.2](#))

Dijital kuantum bilgisayarlarının yanında geliştirilmesi için çaba gösterilen bir başka kuantum bilgisayar türü ise analog kuantum bilgisayarlarıdır. Analog kuantum bilgisayarlar, örneğin elektronlar veya fotonlar gibi Qubit görevi gören temel parçacıkların belirli düzenler dahilinde birbirleri ile etkileşim kuracak konuma getirilmeleri prensibi ile çalışan termal tavlama cihazlarıdır. Kuantum termal tavlama veri bilimindeki klasik tavlama yöntemi ile benzerlik gösteren bir yöntemdir.



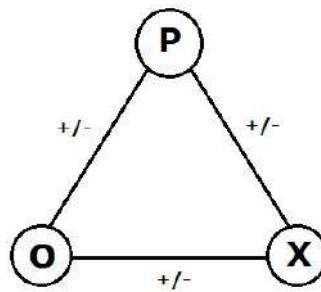
Şekil 1.2. D-Wave Analog Kuantum Bilgisayarı

Amerika Birleşik Devletleri’nde yer alan D-Wave firması analog kuantum bilgisayarlar geliştirmekte ve bu bilgisayarların kullanımını bulut üzerinden kullanıcıllara açmaktadır. ([Sekil 1.3](#)) Çalışmamız dahilinde öncelikle D-Wave’ın bulut kuantum tavlama servislerini kullanarak çeşitli örnek çizgeler üzerinden belli başlı yüneylem problemlerine çözüm getirmeye çalıştık. Bu problemlere örnek olarak Maximum Cut, Map Coloring, Traveling Salesperson, Social Graph ve Clique gösterilebilir. Kuantum tavlama servislerini kullanarak oluşturduğumuz çözümleri klasik işlemciler kullanılarak elde ettiğimiz çözümler ile de kıyaslayarak klasik ve kuantum bilgisayarlarının performans farklarını raporlandırdık.

Kuantum bilgisayarlar henüz tam anlamıyla etkin çalışmadıklarından, performanslarını gerçek yüneylem problemleri üzerinde test etmek güçleşmektedir. Bu sorunu çözmek amacı ile D-Wave bulut üzerinde sunduğu servislerde sadece Kuantum İşlem Birimlerinin (QPU) kullanılabilirliği seçenekten yanında bir de Klasik CPU’ların ve QPU’ların hibrit olarak çalışabildiği bir seçenek de sunmuştur. Çalışmamızın sonraki aşamasında söz konusu yüneylem problemlerine hibrit çözümleyiciler ile yaklaşılmış; ve elde edilen performanslar yine salt CPU çözümleyicilerin performansları ile kıyaslanmıştır.

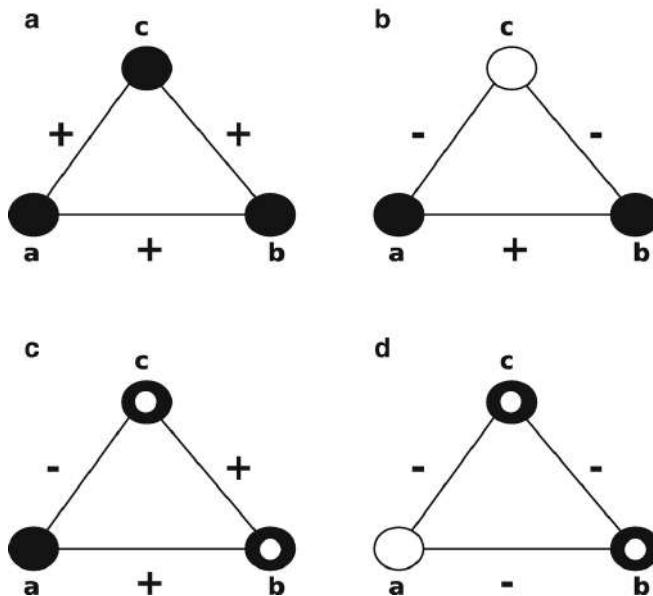
Ardından, projemizin esas konusu olan İşaretli Sosyal Çizge problemine dayalı Yapısal Dengesizlik problemine ilişkin örnek problemlerin oluşturulması yönünde çalışmalar gerçekleştirilmiş; gerçek dünyaya uygun problemler yaratılması için çalışılmıştır. Sonraki aşamada QPU, CPU ve QPU/CPU hibrit işlem birimleri ile bu problemler çözümlenmiş ve ulaşılan sonuçlar performans gösterdikleri süreler ile birlikte raporlanmıştır. Örnek problemlerin yaratımı sırasında dikkate alınan temel faktörler, İşaretli Sosyal Çizge probleminin günlük hayatı kullanılmاسının avantajlı olacağı alanlardır. Bu alanlara örnek gösterilebilecekler; lojistik, asayiş, iç/dış güvenlik, toplumsal ve sosyal düzendir.

İşaretli sosyal çizgeler, her bir noktası bir bireyi veya birimi temsil eden, ve noktalar arasındaki bağlantıların birimler arasındaki pozitif ve negatif ilişkileri ifade ettiği çizge biçimleridir. Sosyal çizgelerde birimler arasındaki olumlu ilişkiler 1 ile ifade edilirken, negatif ilişkiler -1 ile ifade edilmektedir. ([Sekil 1.4](#))



Şekil 1.4. İşaretli Sosyal Çizge

İşaretli sosyal çizgeler sosyal psikoloji, spin camları, kompleks sistemler ve veri kümeleme gibi birçok farklı discipline yönelik konuda konusunda avantaj taşıyan yapılardır. [\[19\]](#) [\[20\]](#) [\[21\]](#) [\[22\]](#) İşaretli sosyal çizgeler söz konusu olduğunda cevaplanması önem taşıyan en önemli soru; çizgenin dengeli durumda olup olmadığıdır. Bu durumun ne anlama geldiğini şıkları üzerinden daha etkin açıklayabiliriz.



Şekil 1.5. Üç birimden oluşan Sosyal Çizge örnekleri

[Şekil 1.5'te](#) yer alan ve üç birimden oluşan Sosyal Çizgeleri inceleyelim. (a) çizgesinde yer alan (a, b), (b, c) ve (a, c) ilişkilerinin tamamı olumlu yöndedir. Bu durumda olası bir gruplaşma durumunda a, b ve c birimleri birbiri ile aynı takımdadır ve doğal olarak bu çizge kendi içinde stabil ve dengeli bir çizgedir. (b) çizgesinde ise (a, b) ilişkisi olumlu yöndeyken (a,c) ve (b,c) ilişkileri negatif konumlanmıştır. Bu durumda olası bir gruplaşmada a ve b birimleri, c birimi ile farklı takımlarda yer alacaktır. Yapısal olarak değerlendirdiğimizde ise yine dengeli bir çizge söz konusudur. (c) çizgesini incelediğimizde ise (a,b) ve (b,c) ilişkileri

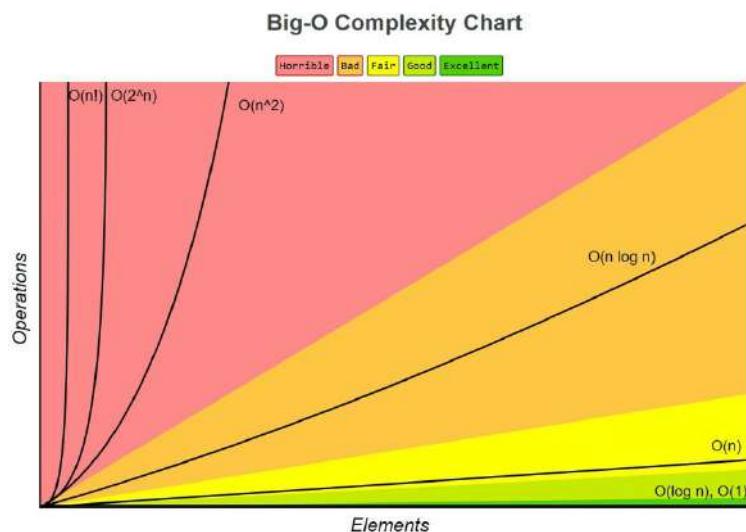
olumluyken (a,c) ilişkisinin olumsuz olduğunu gözlemleriz. O halde bu bireyler gruplandırılırken olası bir takımlaşmada ekibe dahil olan birimlerden c'nin a ile negatif ilişki göstermesine karşın, aynı ekipteki b birimi a ile pozitif ilişki halindedir. Bu durum (c) çizgesini yapısal olarak dengesiz bir çizgeye dönüştürmektedir. Aynı durum, aynı ekibin içerisinde bulunmalarına karşın aralarında negatif ilişki bulunan (d) çizgesinde de gözlemlenebilir.

Sosyal psikoloji üzerine yapılan araştırmalar, sosyal bir çevreyi işaretli çizge şeklinde ifade ettiğimiz durumda olası bir dengesizliğin, o sosyal çevrenin stabil olmadığına yönelik bir kanıt olduğunu öne sürmektedir. ^[23] Dengeli durumda olmayan sosyal çevreler, daha dengeli bir duruma geçebilmek için çeşitli değişiklikler yaşırlar. Bu değişiklikler; negatif ilişkilerin pozitif ilişkilere dönüştüğü olumlu yönde değişim, pozitif yönde ilişkilerin negatif ilişkilere dönüştüğü negatif yönde değişim ve grup değişimi veya grup artışı olabilir.

Yapısal dengesizlik problemini anlayabilmek için popüler kültür üzerinden verilebilecek en yakın durum aşk üçgenleridir. Bu durum aşağıdaki metafor ile açıklanabilir:

1. *Alice* ve *Bob* iki sevgilidir.
2. *Bob, Cassandra* isimli bir kadına ilgi duymaktadır.
3. Haliyle *Alice* ve *Cassandra* birbirlerini sevmemektedir.
4. Bu durumdan dolayı bu üç birimin etkileşiminde yapısal dengesizlik problemi ortaya çıkmaktadır.
5. Sorunun çözümü aşağıdaki şekillerde gerçekleştirilebilir:
 - a. *Alice* ve *Bob* ayrılır. (*Alice-Bob* ilişkisi negatife dönüşür.)
 - b. *Bob, Cassandra* ile görüşmeyi bırakır. (*Bob-Cassandra* ilişkisi negatife dönüşür.)
 - c. *Alice* ve *Cassandra* kendi aralarında anlaşıp *Bob*'u terk ederler. (*Alice-Bob* ve *Bob-Cassandra* ilişkisi negatife dönüşürken, *Alice-Cassandra* ilişkisi pozitife dönüşür.)
 - d. *Alice* ve *Cassandra*, *Bob* ile görüşmeye devam eder ve birbirlerini sevmemeyi bırakırlar. (*Alice-Cassandra* ilişkisi pozitife dönüşür.)

Yukarıda belirtilen örnektenden anlaşılabileceği üzere işaretli sosyal çizgelerde karşılaşılan yapısal dengesizlik problemlerinin birden fazla çözümü olabilmektedir. Ancak örnekteki gibi üç birime sahip bir çizgenin çözümlenmesi kolay olabilirken, çizgeye dahil olan birim sayısı arttıkça ilişkilerin sayısı $\frac{n(n-1)}{2}$ ifadesine bağlı olarak artmaktadır. Bu durum da problemin zaman karmaşıklığını büyük problemler için sorun teşkil edebilecek olan $O(n^2)$ kılmaktadır. ([Sekil 1.6](#))



Sekil 1.3. Büyük-O Zaman Karmaşıklık Grafiği

Yapısal dengesizlik problemlerinin günlük hayatı kullanılabileceği durumlara ilişkin verilebilecek örnekler terör örgütlerinin içindeki yapısal dengesizliklerin tespit edilebilmesi ve örgütü ortadan kaldırılmaya yönelik daha etkin stratejilerin geliştirilebilmesi, toplum ve diğer organizasyon türleri içerisindeki ilişkilerin gözlemlenerek olası yapısal dengesizliklerin çözümlenmesi yoluyla daha sağlıklı ekosistemlerin elde edilmesi gösterilebilir.

Yapısal dengesizlik problemlerinin $O(n^2)$ zaman karmaşıklığına sahip olması mevcut klasik işlemcilerin çok büyük problemler söz konusu olduğunda verimsiz kalmasına sebep olmaktadır. Oysa kuantum bilgisayarlar tasarımları gereği bu problemleri lineer zamanda çözümlerebilecek etkinliğe sahip olabilirler. Çalışmamızın esas amacı D-Wave'in sunduğu Kuantum Tavlama servislerinin İşaretli Sosyal Çizgeler ve yapısal dengesizlik problemi konusundaki fizibilitesini test etmektir. Bu konuya ilgili gerekli örnek problemler tasarlanmış, CPU, QPU ve CPU/QPU hibrit çözümleme araçları ile ölçümlenerek performansları raporlanmıştır.

Amacımız, bu alanda çalışmak isteyen diğer akademik ve endüstriyel çevreler için örnek bir çalışma yaratmak ve analog kuantum bilgisayar teknolojilerinin gerçek dünya problemleri konusunda gelmiş oldukları noktanın fizibilitesini deneysel olarak sınamaktır. Çalışmamız sayesinde bu alanda çalışacak diğer çevreler için teşvik edici olmayı ve alanda daha niş ve detaylı olarak yöneylem problemlerini çözümleyecek uygulamaların geliştirilmesini hızlandırmayı hedefliyoruz.

2. LİTERATÜR ÇALIŞMASI

Kuantum bilgisayarları ve kuantum algoritma geliştirme konusunda gerçekleştirilen çalışmalar 1980'lerde Paul Benioff'un geliştirdiği kuantum Turing makinesi alternatif ile başlamıştır. Sonrasında Peter Shor'un geliştirdiği kuantum tam sayı faktörizasyon algoritması ile devam eden bu süreç, klasik bilgisayarların fizikselli olarak performanslı olmadığı birinci jenerasyon dönemlerdeki algoritma geliştirme sürecine benzetilebilir. Fakat kuantum algoritmalar söz konusu olduğunda işler biraz zorlaşmaktadır. Bunun temel sebebi kuantum algoritmaların çoğunlukla sezgiselliğe uzak olan doğasının yarattığı karmaşıklıktır. Shor'un çalışması haricinde, kuantum algoritmaları üzerine yapılmış çalışmalara örnek olarak Deutsch-Jozsa, Bernstein-Vazirani, Simon algoritması örnek olarak gösterilebilir. [\[24\]](#) [\[25\]](#) [\[26\]](#)

Deutsch-Jozsa algoritması, bir kapalı kutu probleme ilişkin çözüm getirmekte olup; normal koşullarda klasik bilgisayarlar tarafından çok sayıda soru ile çözümlenen bir problemin, bir kuantum bilgisayar ile tek sorguda çözümlenmesini sağlamaktadır. Bernstein-Vazirani algoritması ise, kuantum algoritmaların mevcut bir problemi klasik algoritmardan daha etkin çözümü görülen ilk algoritmadır. Simon algoritması, bir problemin çözümünün kuantum algoritma sayesinde klasik algoritmalarдан üssel seviyede daha hızlı gerçekleştirmesini sağlayan bir çözüm yöntemi ifade etmektedir.

Kuantum algoritmalar ek olarak kuantum bilgisayarların olası donanımsal özellikleri üzerine gerçekleştirilen çalışmalar da mevcuttur. Bu çalışmalara örnek olarak Google, IBM ve Microsoft'un araştırma geliştirme sürecinde oldukları dijital kuantum bilgisayarlarında gelmiş oldukları noktaya yönelik paylaştıkları akademik çalışmalara ek olarak; dünyanın farklı bölgelerinde düşük sıcaklık laboratuvarlarında süper iletkenler üzerine çalışan üniversitelerin gerçekleştirdiği çalışmaları gösterebiliriz. [\[27\]](#) [\[28\]](#) Bu çalışmaların odaklandıkları noktalar; kuantum bilgisayarların geliştirilmesi için önem taşıyan çeşitli farklı disiplinlere yönelikdir. Bu disiplinlere örnek olarak mikrodalga fiziği, materyal mühendisliği, donanım mühendisliği, kuantum mühendisliği gösterilebilir.

Bunun yanında, D-Wave'in sunduğu kuantum tavlama bulut servislerine yönelik akademik çalışmalar da gerçekleştirilmiştir. Bu çalışmalar da gerçekleştirdiğimiz çalışmaya benzer olarak D-Wave'in sunduğu tavlama servislerini denemeye ve mevcut klasik CPU'lar ile performanslarını kıyaslamaya, belli noktalarda fizibilitelerini test etmeye yönelikir. [\[29\]](#) [\[30\]](#)

Çalışmamızın diğerlerinden farklı yönü, temel olarak işaretli sosyal çizgeler ve yapısal dengesizlik probleminin D-Wave kuantum tavlama servisleri kullanılarak gerçek dünyada karşılaşabilecek durumlara ilişkin fizibilitesinin incelenmesine odaklanmasıdır. Bu alanda gerçekleştirdiğimiz çalışma ile, konu hakkında daha niş ve detaylı akademik ve endüstriyel çalışmaları teşvik etmeyi ve önünü açmayı amaçlıyoruz.

3. YÖNTEM VE TEKNOLOJİLER

Çalışmamızda D-Wave kuantum bulut sistemlerini denerken daha açık ve kolay ifade edilebilen bir dil tercih etme amacıyla Python 3'ü seçtik. Çalışma dahilinde gerçekleştirilen sıralı işlemlerin daha düzenli görüntülenebilmesi amacıyla Jupyter Notebook üzerinde Markdown özelliğini aktif olarak kullanarak deneylerimizi gerçekleştirdik. Çalışmamızda kullandığımız çeşitli tiplerdeki örnek çizgeleri yaratmak için NetworkX ve D-Wave NetworkX kütüphanelerinden faydalandık. D-Wave'in kuantum tavlama bulut servislerine erişmek için Ocean SDK kütüphanesini kullandık. Leap servisi üzerinden sağlanan API anahtarı ve konfigürasyon dökümanları sayesinde bulut servis ile uzak bağlantımızı gerçekleştirdik. Çizgeler, tablolar vb. görselleştirmeler için matplotlib kütüphanesinden faydalandık. Çalışmamızda incelediğimiz yüneylem problemleri Clique, Maximum Cut, Map Coloring, Traveling Salesman, Structural Imbalance olarak belirlenmiştir ve ana konu olarak İşaretli Sosyal Çizgeler üzerindeki Yapısal Dengesizlik problemi belirlenmiştir.

Çalışmamızda kuantum tavlama bulut servislerinin tercih edilmesinin temel sebebi bu teknolojilerin henüz yeni geliştirilmekte olan ve gerek akademik, gerek endüstriyel çevrelerce faydası henüz tam olarak test edilmemiş ve fizibilitesi konusunda net bir kanya varılmamış bir noktada olmalarıdır. Bu durumda, bu alanların olası faydalarının ve performanslarının akademik çalışmalarca belgelenmesi, daha ileri düzeydeki çalışmaların önünü açabilir ve alandaki çalışmaların artması ile birlikte yeni kullanım alanlarının ortaya çıkışmasını sağlayabilir. Kuantum bilgisayarlarının gelecekte yüneylem alanındaki optimizasyon problemlerinde avantaj sağlayabilecek olması, çalışmada bu teknolojilerin tercih edilmesinin temel sebeplerinden bir diğeridir. Önümüzdeki on yıllar içerisinde alandaki bilgi düzeyi arttıkça başta lojistik, sağlık, ekonomi ve güvenlik olmak üzere birçok alanda kuantum bilgisayarların yarattığı faydanın, genişleyen kullanım alanları ile birlikte daha aktif duruma geçeceğini düşünmektediriz.

4. ANALİZ VE PROBLEM TASARIMI

Çalışmada yer alan temel yüneylem problemlerini aşağıdaki gibi özetleyebiliriz:

1. Clique
2. Map Coloring
3. Vertex Cover
4. Matching
5. Maximum Cut
6. Signed Social Graph
7. Traveling Salesperson

Bahsedilen yüneylem problemlerinin, örnek çizgeler üzerinden CPU, QPU ve CPU/QPU hibrit sistemler kullanılarak çözümlenmesi ve performanslarının ölçümlenmesi gerekmektedir. Bu ölçümlemeler ve performans değerlendirmeleri sonucunda ortaya çıkan sonuçların uygun şekilde raporlanmasıının ardından yapısal dengesizlik problemine ilişkin çalışılması amaçlanmaktadır.

Yapısal dengesizlik problemine ilişkin birden fazla sayıda örnek problem çizgeler üzerinden tanımlanacak ve problem tanımları gerçekleştirilecektir. Ardından çizgelerin yazılım boyutundaki ifadeleri ve görselleştirmeleri gerçekleştirilecektir. Bu süreçte de aynı şekilde bahsedilen problemlerin CPU, QPU ve CPU/QPU hibrit sistemleri kullanılarak ölçümlenmesi ve performans değerlendirmelerinin yapılması hedeflenmektedir. Üretilen sonuçların etkin bir şekilde raporlanması önem taşımaktadır.

Yapısal dengesizlik problemlerine ilişkin değerlendirmelerin raporlanmasıının ardından CPU ve QPU sistemleri arasındaki performans farkının mevcut durumu değerlendirilecek, günümüz endüstriyel uygulamalarında probleme ilişkin kullanım alanları hakkında tartışılabilecek ve bu alandaki fizibilite durumu konusunda fikirlerden bahsedilecektir.

Çalışmamızı gerçekleştirirken görselleştirmeleri ve sıralı deneysel çalışmaları daha iyi ifade edebilmek amacıyla Jupyter Notebook kullanımını daha verimli bulduk. Bu ortamda gerçekleştirdiğimiz sıralı çalışmaları Markdown özelliği sayesinde çeşitli notlar ile işaretledik ve adım adım belirttik.

Gerçekleştirilen deneysel çalışmaların çalışma dahilinde incelenen problem türlerine göre düzenli bir şekilde işlenmesi ve performanslarının ayrı ayrı test edilmesi çalışmanın kalitesi açısından önem taşıyacağından bu konuya ilgili azami dikkat gösterdik ve anlaşılmabilirliği yüksek tutmaya çalıştık.

Yapısal dengesizlik problemi ana konu olmak üzere; Map Coloring, Clique, Maximum Cut, Traveling Salesman gibi belli başlı problemlerin alt başlıklar halinde incelenebildiği bir notebook ortamı kullandık ve her bir problemin çizge şeklindeki ifadesini görselleştirdikten sonra CPU, QPU ve CPU/QPU hibrit servisleri ile ayrı ayrı performanslarının raporlanıldığı alt başlıklar oluşturmayı tasarladık.

Deneysel olarak sistemin performansının ölçümlendiği yönyelem problemlerinden sonra; çalışmanın ana konusu olarak işaretlenecek sosyal çizgелere bağlı Yapısal Dengesizlik problemine yönelik oluşturulabilecek etkin ve gerçeğe yakın alt problemlerin oluşturulması konusunda tasarılar gerçekleştirdik. Bu problemler notebook dahilinde çeşitli alt başlıklarda değerlendirilecek, açık bir şekilde tanımlanacak, ardından yine CPU, QPU ve CPU/QPU hibrit servisleri ile ayrı ayrı çözümlenip performans değerleri ayrı ayrı raporlanarak gerekli görülen bölgelerde tablolar ve grafikler ile görselleştirilecek.

5. NETWORK-X KÜTÜPHANESİ İLE ÇİZGE OLUŞTURMA

Çalışmamızın bu bölümünde Network-X kütüphanesi kullanılarak, yaratılabilecek örnek çizge problemlerini gözden geçireceğiz. Network-X, ücretsiz ve açık kaynak olarak kullanılabilen, ağırlıklandırılmış veya homojen bağlantı noktaları ve nodlardan oluşan çeşitli çizgeler üretmeye ve işlemeye olanak sağlayan, Python 3 ortamına uyumlu bir kütüphanedir.

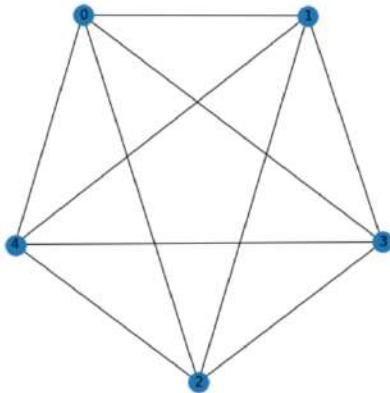
Network-X kütüphanesi beraberinde bulunan Graph Generator fonksiyonları dahilinde hazır çizgeler üretebilen bir kaynaktır. Bu fonksiyonlar sayesinde birçok farklı çizge türü yazılımsal olarak kolayca üretilibilmektedir. Network-X Graph Generator fonksiyonları ile üretilebilecek çizgelere aşağıdaki örnekleri verebiliriz:

- Complete Graph
- Balanced Tree Graph
- Barbell Graph
- Circular Ladder Graph
- Cycle Graph
- Ladder Graph
- Lollipop Graph
- Star Graph
- Turan Graph
- Wheel Graph

Graph Generator sınıfının hazır fonksiyonlarına ek olarak node'larını ve kenar ağırlıklarını kendimiz tanımladığımız özel grafikleri de Network-X kütüphanesi kullanarak kolayca ifade edebilmemiz mümkündür. Bölümün ilerleyen kısımlarında Network-X kütüphanesi kullanılarak üretilmiş örnek hazır çizgeleri ve özel olarak tanımlanmış ağırlıklara sahip çizgelerin oluşturulma basamaklarını inceleyebilirsiniz.

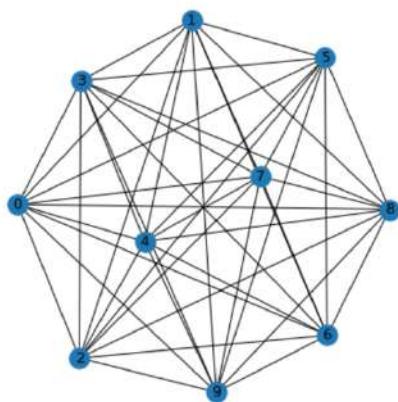
Complete Graphs

```
In [7]: cg1 = nx.complete_graph(5)
plt.figure(figsize=(5,5))
nx.draw(cg1, with_labels=True, font_weights='bold')
```



Şekil 5.1. Network-X. Örnek Complete Graph 1

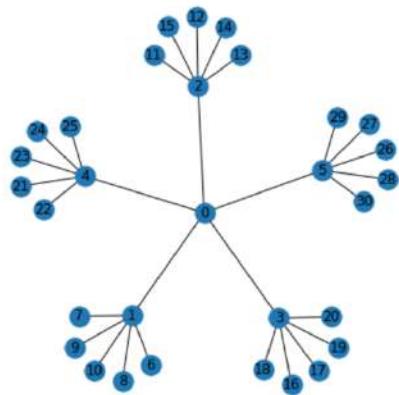
```
In [8]: cg2 = nx.complete_graph(10)
plt.figure(figsize=(5,5))
nx.draw(cg2, with_labels=True, font_weights='bold')
```



Şekil 5.2. Network-X. Örnek Complete Graph 2

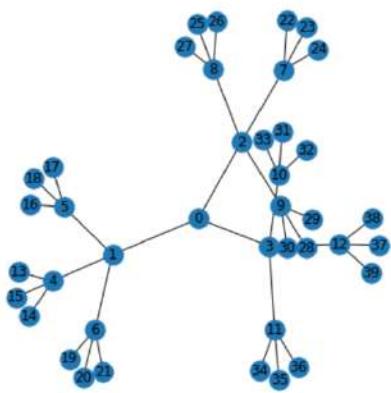
Şekil 5.1 ve Şekil 5.2'de incelediğimiz Tamamlanmış Çizgeler, Network-X kütüphanesi kullanılarak oluşturulmuştur. Tamamlanmış Çizgelerdeki tüm node'lar, diğer node'lar ile birer adet bağlantı içerisinde bulunmaktadır ve eksik bağlantı yer almamaktadır. Şekil 5.1'deki çizge 5 adet node'dan meydana gelirken, Şekil 5.2'deki çizge 10 adet node'dan meydana gelmektedir.

```
In [9]: bt1 = nx.balanced_tree(5, 2)
plt.figure(figsize=(5,5))
nx.draw(bt1, with_labels=True, font_weights='bold')
```



Şekil 5.3. Network-X Örnek Balanced Tree 1

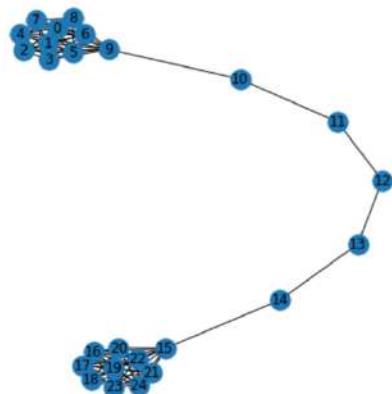
```
In [10]: bt2 = nx.balanced_tree(3, 3)
plt.figure(figsize=(5,5))
nx.draw(bt2, with_labels=True, font_weights='bold')
```



Şekil 5.4. Network-X Örnek Balanced Tree 2

Şekil 5.3 ve Şekil 5.4'de Network-X kütüphanesi kullanılarak oluşturulmuş hazır Balanced Tree örneklerini inceleyebiliriz. Network-X'in ilgili metodunu kullanırken kullandığımız ilk parametre ağacın dallanma faktörünü temsil ederken, ikinci parametre ise ağacın yüksekliğini temsil etmektedir. Şekil 5.3'da 5'er daldan oluşan ve toplamda 2 katmanlı bir ağaç gözlemlerken, Şekil 5.4'de ise her katta 3'er dallanmadan oluşan 3 katmanlı bir ağaç gözlemliyoruz.

```
In [11]: bg1 = nx.barbell_graph(10, 5)
plt.figure(figsize=(5,5))
nx.draw(bg1, with_labels=True, font_weights='bold')
```



Şekil 5.5. Network-X. Örnek Barbell Graph 1

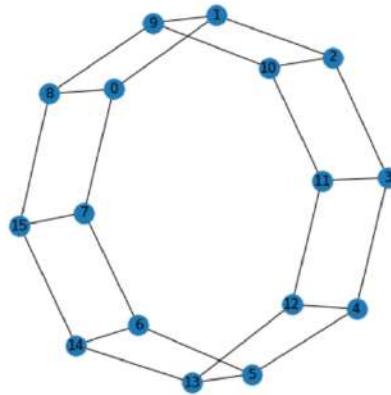
```
In [12]: bg2 = nx.barbell_graph(8, 15)
plt.figure(figsize=(5,5))
nx.draw(bg2, with_labels=True, font_weights='bold')
```



Şekil 5.6. Network-X. Örnek Barbell Graph 2

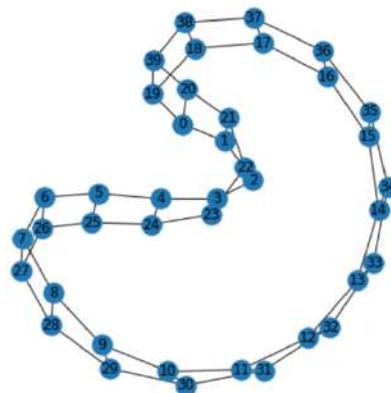
Şekil 5.5 ve Şekil 5.6'da Network-X kütüphanesi kullanılarak üretilmiş Barbell Çizgelerini inceleyebiliriz. Bu çizgeleri oluşturan ilgili metot kullanılırken yerleştirilen ilk parametre çizgenin iki ucunda yer alacak birbirlerine bağlı node sayısını ifade ederken, ikinci parametre bu iki node yiğinını birbirine bağlayan yoldaki sıralı node'ların toplam sayısını ifade etmektedir. Şekil 5.5'de birbirine bağlı 10 adet node'dan oluşan iki yiğinını birbirine bağlayan 5 node gözlemlerken, Şekil 5.6'da 8'er adetlik iki node yiğinını birbirine bağlayan 15 node gözlemliyoruz.

```
In [13]: c11 = nx.circular_ladder_graph(8)
plt.figure(figsize=(5,5))
nx.draw(c11, with_labels=True, font_weights='bold')
```



Şekil 5.7. Network-X Örnek Circular Ladder Graph 1

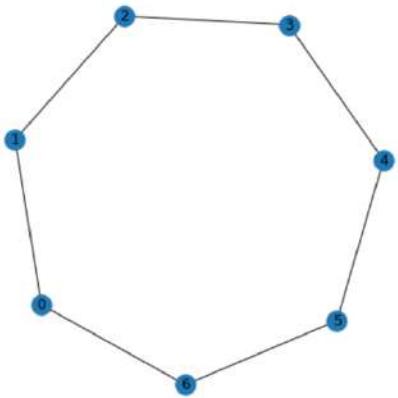
```
In [14]: c12 = nx.circular_ladder_graph(20)
plt.figure(figsize=(5,5))
nx.draw(c12, with_labels=True, font_weights='bold')
```



Şekil 5.8. Network-X Örnek Circular Ladder Graph 2

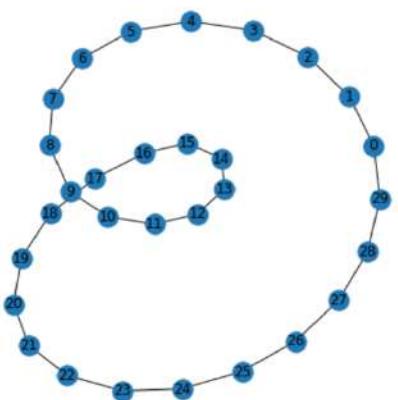
Şekil 5.7 ve Şekil 5.8'de Network-X kütüphanesi kullanılarak üretilmiş Circular Ladder Çizgelerine örnekler gözlemliyoruz. Şekil 5.7'deki çizgede 8 basamaklı ve toplamda 16 adet node'dan oluşan bir çizge meydana getirildi. Bu çizgeler isimlerini merdiveni anımsatmalarından edinmişlerdir ve eşlerine ve komşularına bağımlı node'lardan oluşmaktadır. Şekil 5.8'de ise 20 adet merdiven basamağından oluşan toplamda 40 adet node'un meydana getirdiği çizgeyi gözlemliyoruz.

```
In [15]: cyg1 = nx.cycle_graph(7)
plt.figure(figsize=(5,5))
nx.draw(cyg1, with_labels=True, font_weights='bold')
```



Şekil 5.9. Network-X. Örnek Cycle Graph 1

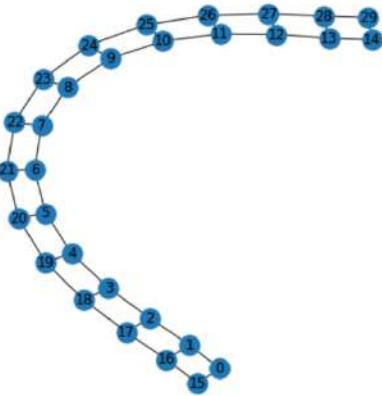
```
In [16]: cyg2 = nx.cycle_graph(30)
plt.figure(figsize=(5,5))
nx.draw(cyg2, with_labels=True, font_weights='bold')
```



Şekil 5.10. Network-X. Örnek Cycle Graph 2

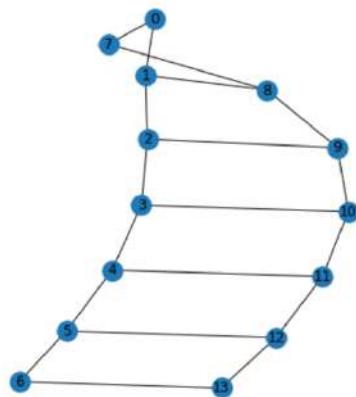
Şekil 5.9 ve Şekil 5.10'da Network-X kütüphanesi kullanılarak oluşturulan örnek halkasal çizgeleri inceliyoruz. Bu çizgeler oluşturulurken kullanılan ana parametre toplamda kaç adet node içereceklerinin bilgisidir. Bu çizge tipini ayırt eden nokta ise, çizgedeki her node'un solundaki ve sağındaki bir node ile bağlantı kuruyor olması ve doğal olarak halkasal şekilde ifade edilmesidir.

```
In [17]: lag1 = nx.ladder_graph(15)
plt.figure(figsize=(5,5))
nx.draw(lag1, with_labels=True, font_weights='bold')
```



Şekil 5.11. Network-X Örnek Ladder Graph 1

```
In [18]: lag2 = nx.ladder_graph(7)
plt.figure(figsize=(5,5))
nx.draw(lag2, with_labels=True, font_weights='bold')
```



Şekil 5.12. Network-X Örnek Ladder Graph 2

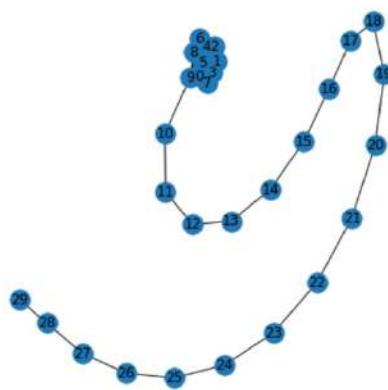
Şekil 5.11 ve Şekil 5.12'de Network-X kütüphanesi kullanılarak üretilmiş Ladder Çizgelerini inceliyoruz. Bu çizgelerin Circular Ladder çizgelerden en temel farkı halkasal yapıda olmamalarıdır. Doğal olarak bu çizgeler, yine bir merdiven görünümünde olup, karşılarında bulunan nodelar ve komşu node ile bağlantı halinde iken baş ve sonlardaki node'larda bu merdiven sonlanmaktadır. Oysa Circular Ladder çizgelerde sonda bulunan node'lar yeniden başta bulunan node'lar ile birleşmekte ve bu halkasal bir görünüm meydana getirmektedir. Bu çizgeler oluşturulurken kullanılan temel parametre yine basamak sayısıdır.

```
In [19]: log1 = nx.lollipop_graph(10, 5)
plt.figure(figsize=(5,5))
nx.draw(log1, with_labels=True, font_weights='bold')
```



Şekil 5.13. Network-X Örnek Lollipop Graph 1

```
In [20]: log2 = nx.lollipop_graph(10, 20)
plt.figure(figsize=(5,5))
nx.draw(log2, with_labels=True, font_weights='bold')
```

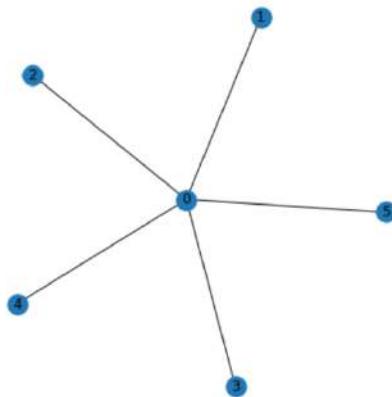


Şekil 5.14. Network-X Örnek Lollipop Graph 2

Şekil 5.13 ve Şekil 5.14'de Network-X tarafından oluşturulmuş örnek Lollipop çizgelerini görmekteyiz. Bu çizgeler de yapısal olarak Barbell tipi çizgelerle benzerlik göstermektedir ancak Barbell çizgelerde tek ucta bulunan node yiğitleri Lollipop tipi çizgelerde tek tarafta bulunmaktadır. Ve bu yiğiti sıralı node'lar takip etmektedir. Bu tür çizgeleri Network-X kullanarak oluşturmak istedigimizde kullandığımız ilk parametre yiğitta bulunacak node sayısını ifade ederken, ikinci parametre ise yiğiti sıralı takip eden node'ların kaç adet olacağını ifade etmektedir. Örneğin Şekil 5.13'da 10 adet node'un bulunduğu yiğiti takip

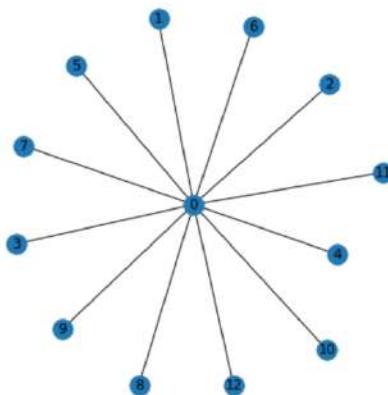
eden 5 adet node bulunmaktadır. Şekil 5.14'deki yığitta ise yine 10 adet node olmasına karşın, takip eden 20 adet sıralı node bulunmaktadır.

```
In [21]: stg1 = nx.star_graph(5)
plt.figure(figsize=(5,5))
nx.draw(stg1, with_labels=True, font_weights='bold')
```



Şekil 5.15. Network-X Örnek Star Graph 1

```
In [22]: stg2 = nx.star_graph(12)
plt.figure(figsize=(5,5))
nx.draw(stg2, with_labels=True, font_weights='bold')
```



Şekil 5.16. Network-X Örnek Star Graph 2

Şekil 5.15 ve Şekil 5.16'da Network-X kütüphanesi kullanılarak oluşturulmuş Yıldız çizgeleri gözlemliyoruz. Bu çizgelerin temel özelliği parametre olarak verilen N adet node'un bir adet merkezi node etrafına yerleştirildiği ve onunla bağlantılılığı mimariyi ifade ediyor olmalarıdır. Şekil 5.15'deki çizgede 0 merkezi node'unun etrafına yerleştirilen, fonksiyon parametrelerinde ifade edildiği üzere 5 adet node gözlemliyoruz. Şekil 5.16'daki

çizgede ise yine merkeze yerleştirilen 0 numaralı node'un etrafına yerleştirilen 12 adet node gözlemliyoruz.



Şekil 5.17. Network-X Örnek Turan Graph 1

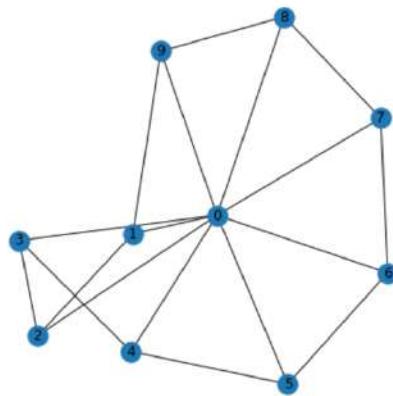


Şekil 5.18. Network-X Örnek Turan Graph 2

Şekil 5.17 ve Şekil 5.18'de Network-X kütüphanesi kullanılarak oluşturulmuş örnek Turan Çizgeleri gözlemliyoruz. Bu çizgelerin özelliği ise, ilk parametrelerinde belirtilen sayıda node içermeleridir. İkinci parametresi ise ilk parametresinden daha az veya eşit olmak koşulu ile, çizgede yer alan node'ların

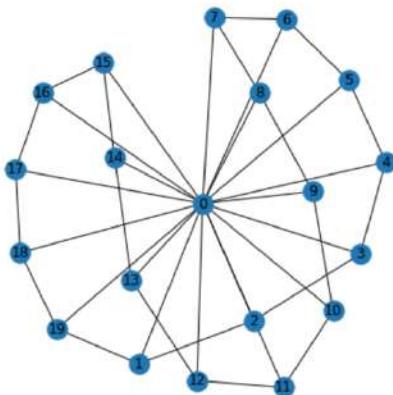
alt kümelerine yönelik bölüntülenme miktarını ifade etmektedir. Harita boyama problemlerinde kullanım örneklerine rastlanabilen bir çizge türüdür.

```
In [25]: whg1 = nx.wheel_graph(10)
plt.figure(figsize=(5,5))
nx.draw(whg1, with_labels=True, font_weights='bold')
```



Şekil 5.19. Network-X Örnek Wheel Graph 1

```
In [26]: whg2 = nx.wheel_graph(20)
plt.figure(figsize=(5,5))
nx.draw(whg2, with_labels=True, font_weights='bold')
```



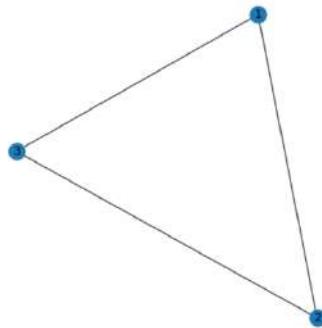
Şekil 5.20. Network-X Örnek Wheel Graph 2

Şekil 5.19 ve Şekil 5.20'de Network-X kütüphanesi kullanılarak oluşturulmuş Tekerlek çizgelere örnekler göremekteyiz. Bu çizgeler parametrelerinde belirtilen sayıda node'dan oluşmak ile birlikte merkez node ile bağlantılı, aynı zamanda kendi komşu node'ları ile de bağlantılı dış node'lardan oluşmaktadır. Dolayısı ile bu çizgeler incelemiş olduğumda tekerleği anımsattıklarından,

isimlerini bu özellikleri sayesinde edinmişlerdir. Şekil 5.19'da 10 adet toplam node'dan meydana gelen bir Tekerlek çizge gözlemlerken, Şekil 5.20'de toplamda 20 adet node'dan meydana gelen bir Tekerlek çizge inceleyebiliriz.

```
In [23]: spec_graph = nx.Graph()
spec_graph.add_edge(1,2, weight=7)
spec_graph.add_edge(2,3, weight=9)
spec_graph.add_edge(3,1, weight=17)

plt.figure(figsize=(5,5))
nx.draw(spec_graph, with_labels=True, font_weights='bold')
```



Şekil 5.21. Network-X Örnek Özel Ağırlıklı Çizge

Sadece hazır çizge tiplerini oluşturabilmek dışında, Network-X kütüphanesi kullanılarak özel ağırlıklara sahip çizgeler de tanımlanabilmektedir. Bunu gerçekleştirebilmek için öncelikli olarak Network-X'in Graph sınıfını kullanarak bir çizge objesi yaratmamız gerekmektedir. Ardından çizgeye ekleyeceğimiz kenarları, bağladıkları node'lar ve ağırlık bilgilerini de vererek ilgili add_edge metodunu kullanarak yaratmalıyız. Şekil 5.21'de gözlemlediğimiz çizgede 3 adet node bulunmaktadır. Network-X'in Graph sınıfına ait add_edge metodunu kullanarak bu çizgeye (1,2), (2,3) ve (3,1) bağlantılarını sırasıyla 7, 9 ve 17 ağırlıkları ile ekledik. Bu sayede özel ve heterojen ağırlıklara sahip bir çizge oluşturmuş olduk.

Bu bölümde Network-X ile oluşturulan çizgeler görüntülenirken, görselleştirmede Matplotlib.pyplot kütüphanesi kullanılmıştır. Çizgelerin ekranda belirtilen formlarında node'lar açıkça belirtilirken node'ları birbirine bağlayan kenarlardaki ağırlıklar açık olarak gösterilmemektedir.

6. D-WAVE QPU VE CPU İLE ÖRNEK PROBLEMLERİN ÇÖZÜMÜ

Bu bölümde, Network-X kütüphanesi kullanılarak yaratılmış çeşitli örnek çizgeler ile ifade edilen problemler tanımladık. Ardından, yaratılan problemlerin D-Wave Kuantum Tavlama servisleri kullanılarak çözümlenmesi için gerekli metotları kullanarak bu problemlere çözümlemeler getirdik ve performanslarını ölçümledik.

```
In [48]: #CPU Çözümleyicisinin import edilmesi
from dimod.reference.samplers import ExactSolver

#QPU Çözümleyicisinin import edilmesi
from dwave.system.samplers import DWaveSampler

from dwave.system.composites import EmbeddingComposite
import dwave_networkx as dnx

import time
import random
```

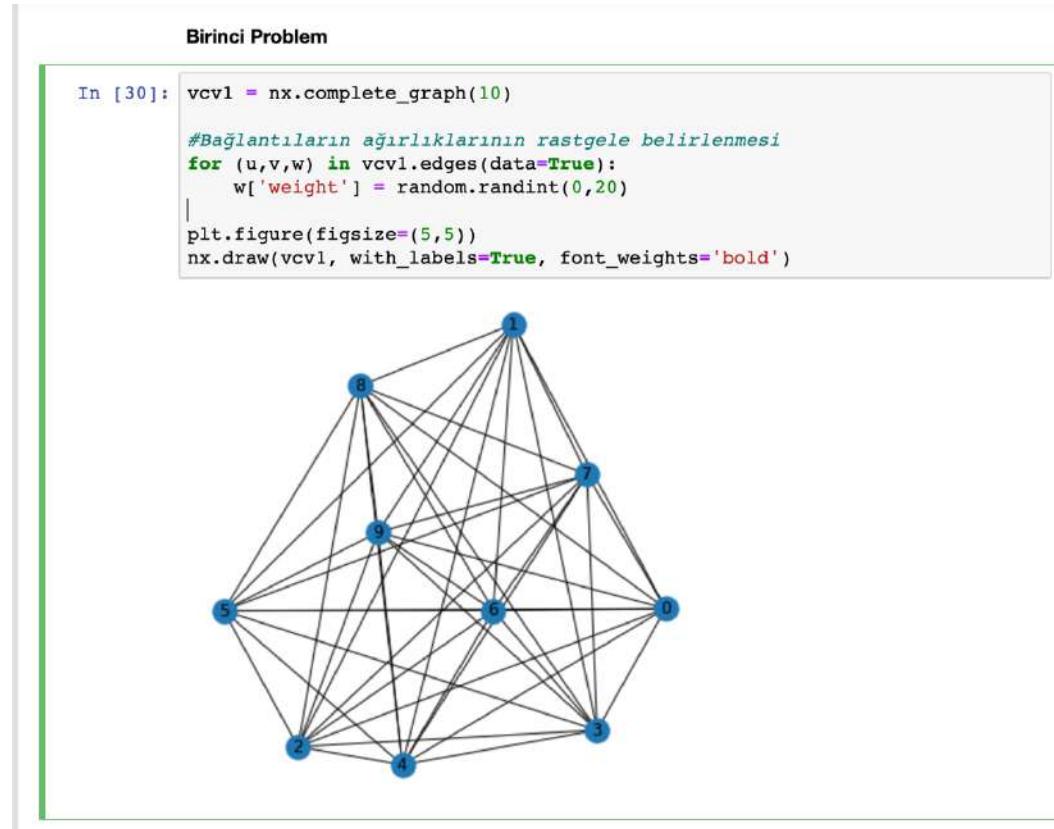
Şekil 6.1. Gerekli kütüphanelerin import edilmesi

Şekil 6.1'de görüleceği üzere çözüm işlemlerinin gerçekleştirilebilmesi için kullanılması gereken belli başlı kütüphaneler bulunmaktadır. Bu kütüphanelerden ExactSolver, problemlere ilişkin çözümlerin klasik CPU ünitesi ile gerçekleştirilmesini sağlamaktadır. Buna karşın, DwaveSampler ise problemlere ilişkin çözümlerin D-Wave'e ait Kuantum Bilgisayarlar tarafından (QPU) çözümlenmesine olanak sağlamaktadır. EmbeddingComposite, problemlerimizi D-Wave'in sahip olduğu Kuantum işlem birimlerinin veri işleme formatı olan QUBO formatında kullanmamıza imkan vermektedir. Ayrıca Network-X kütüphanesinin D-Wave'in kullanım amaçlarına yönelik modifiye edilmiş bir formatı olan D-Wave Network-X kütüphanesini de import ettik. Time ve Random değişkenleri ise ilerleyen bölümlerde gösterilecek olan çeşitli işlem zamanı ölçümleri ve problem tanımları gibi amaçlarla kullanılmak için import edildi.

6.1 Minimum Vertex Cover Problemi

Bu bölümde, Minimum Vertex Cover problemine ilişkin üç adet örnek tanımladık. Bu örneklerin D-Wave'in Kuantum Bilgisayarlarının QPU işlem birimlerine ve klasik CPU'lara dayanarak çözümlerini gerçekleştirdik ve performanslarını değerlendirdik. Problemleri yaratırken çeşitli boyutlardaki çizgeler kullandı ve bunların çözüm performansları üzerinde yarattığı etkiyi kontrol ettik.

Minimum Vertex Cover problemi, verilen bir çizgede her bir bağlantıyı gören en az bir adet birim olacak şekilde çizgenin sahip olabileceği en az sayıdaki birimi bulmayı amaçlayan bir problemdir. Normal koşullarda NP-Complete karmaşıklık sınıfında yer almaktadır. Dolayısı ile geniş birim ve bağlantı miktarları söz konusu olduğunda, klasik bilgisayarlar kullanılarak tam anlamıyla tutarlı cevaplar alınabilmesi zorlu olan problemler arasında yer almaktadır. Tam anlamıyla tutarlı cevaplar istendiğinde ise artan birim ve bağlantı sayısı ile birlikte çözüm süresi üssel olarak büyümektedir.



Şekil 6.1.1. Minimum Vertex Cover, 1. Problem

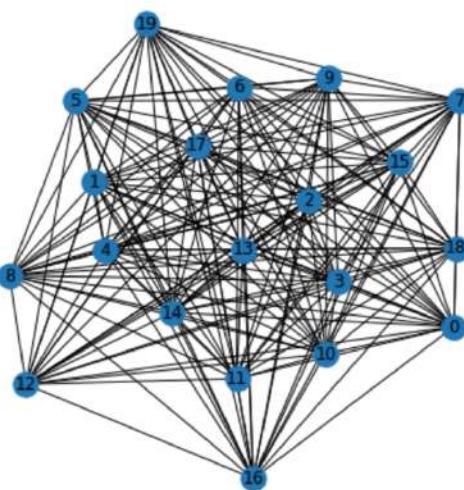
Şekil 6.1.1'de gözlemleneyebildiğimiz Birinci Problem, 10 adet birimden meydana gelen bir tam çizgeyi ifade etmektedir. Bu çizgenin içinde yer alan her birimin birbiri ile olan bağlantısı bulunmaktadır. Bağlantıların ağırlıkları ise, 0 ve 20 arasındaki sayılar ile rastgele olacak şekilde belirlenmiştir. Bu sayede kod satırı her çalıştığında, birbirinden daha farklı niteliklere sahip örnek problemler meydana gelecektir. Çizge yaratıldıktan sonra matplotlib ve Network-X kütüphanesi kullanılarak çıktı şeklinde görüntülenmiştir.

İkinci Problem

```
In [31]: vcv2 = nx.complete_graph(20)

for (u,v,w) in vcv2.edges(data=True):
    w['weight'] = random.randint(0,20)

plt.figure(figsize=(5,5))
nx.draw(vcv2, with_labels=True, font_weights='bold')
```



Şekil 6.1.2. Minimum Vertex Cover, 2. Problem

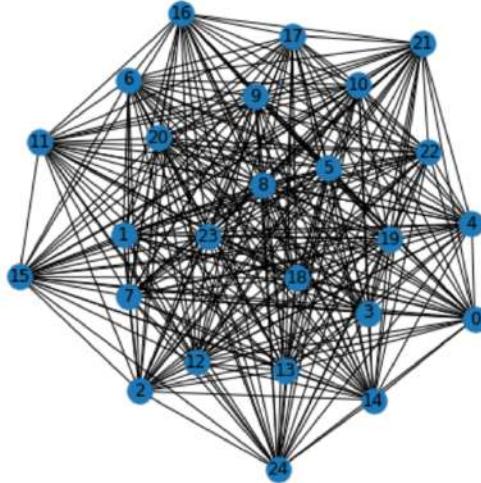
Şekil 6.1.2'de inceleyebileceğimiz İkinci Problem 20 adet birimden meydana gelmektedir. Bu çizgede yer alan bütün birimler de, önceki problemde olduğu gibi birbiri ile bağlantılı durumdadır. Bu problemde birimleri birbirine bağlayan 190 adet bağlantı noktası yer almaktadır. Önceki problemde olduğu gibi bu problemde de, bağlantı noktalarının ağırlıkları 0 ve 20 arasındaki sayılardan seçilecek şekilde rastgele olarak belirlenmiştir.

Üçüncü Problem

```
In [32]: vcv3 = nx.complete_graph(25)

for (u,v,w) in vcv3.edges(data=True):
    w['weight'] = random.randint(0,20)

plt.figure(figsize=(5,5))
nx.draw(vcv3, with_labels=True, font_weights='bold')
```



Şekil 6.1.3. Minimum Vertex Cover, 3. Problem

Son olarak şekil 6.1.3'de gözlemlediğimiz Üçüncü Problemde 25 birim bulunmaktadır. Bu birimler de önceki problemlerdeki gibi birbiri ile bağlantılıdır. Bağlantılarının ağırlıkları 0 ve 20 arasında rastgele olacak şekilde belirlenmiştir.

Birinci Problem - QPU ile Çözümleme

```
In [33]: q_start = time.time()

qpu_result_a = dnx.min_vertex_cover(vcv1, qpu_sampler)

q_end = time.time()

qpu_result_time = (q_end - q_start)

print("1. PROBLEM - QPU ÇÖZÜM SÜRESİ: ",qpu_result_time)
```

1. PROBLEM - QPU ÇÖZÜM SÜRESİ: 5.6201417446136475

Şekil 6.1.4. Minimum Vertex Cover, 1.Problemin QPU ile Çözümlenmesi

Şekil 6.1.4'de gözlemeleyebileceğimiz gibi, yukarıda bahsedilen Birinci Problemin D-Wave kuantum işlem birimi kullanılarak çözümlenmesi ~5.62

saniyede gerçekleştirilmiştir. Gerçekleştirilen işlemin ne kadar sürdüğü, Python'a ait time kütüphanesi kullanılarak ölçülmüştür.

```
Birinci Problem - CPU ile Çözümleme

In [34]: c_start = time.time()

cpu_result_a = dnx.min_vertex_cover(vcv1, cpu_sampler)

c_end = time.time()

cpu_result_time = (c_end - c_start)

print("1. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)

1. PROBLEM - CPU ÇÖZÜM SÜRESİ: 0.004764080047607422
```

Şekil 6.1.5. Minimum Vertex Cover, 1. Problemin CPU ile Çözümlenmesi

Şekil 6.1.5'de ise, Birinci Problemin klasik CPU kullanılarak çözümlenmesinin ~0.004 saniye süregünü gözlemliyoruz. Bu senaryoda, kuantum işlem birimi, CPU'nun performansının yanında daha yavaş kalmış gibi görünüyor. Bunun en önemli sebeplerinden birisi, D-Wave'in sunduğu kuantum tavlama sistemlerinin bulut üzerinden erişilebilir olmasının yarattığı HTTP istek ve yanıt gecikmeleri olabilir. Ölçüm yapmanın daha kolay olabileceği, kar/zarar noktasını belirlemenin daha rahat gerçekleşeceği bir denge noktası yakalayabilmek adına, problemlerin boyutlarını aşamalı olarak arttırdık.

```
İkinci Problem - QPU ile Çözümleme

In [35]: q_start = time.time()

qpu_result_b = dnx.min_vertex_cover(vcv2, qpu_sampler)

q_end = time.time()

qpu_result_time = (q_end - q_start)

print("2. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)

2. PROBLEM - QPU ÇÖZÜM SÜRESİ: 5.60565972328186
```

Şekil 6.1.6. Minimum Vertex Cover, 2. Problemin QPU ile Çözümlenmesi

Şekil 6.1.6'da İkinci Problemin QPU kullanılarak çözümlenmesinin ~5.6 saniye süregünü görüyoruz. Önceki problemde 45 bağlantı noktası, ikinci

problemde 190 bağlantı noktası bulunmasına rağmen, kuantum işlem biriminin problemi çözme süresinde anlamlı bir değişiklik olmadı.

İkinci Problem - CPU ile Çözümleme

```
In [36]: c_start = time.time()

cpu_result_b = dnx.min_vertex_cover(vcv2, cpu_sampler)

c_end = time.time()

cpu_result_time = (c_end - c_start)

print("2. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)

2. PROBLEM - CPU ÇÖZÜM SÜRESİ: 3.043724775314331
```

Şekil 6.1.7. Minimum Vertex Cover, 2. Problemin CPU ile Çözümlenmesi

Şekil 6.1.7'de İkinci Problemin klasik CPU ile çözümlenmesinin ~3.04 saniyede gerçekleştiğini gözlemliyoruz. Birinci problemin ~0.004 saniyede gerçekleştiğini düşündüğümüzde çözüm süresinde ciddi bir artış olduğunu söyleyebiliriz. Bu durum,问题in klasik CPU'lar üzerindeki fizibilitesinin sınırlarına yaklaştığımızın bir göstergesi olarak değerlendirilebilir.

Üçüncü Problem - QPU ile Çözümleme

```
In [37]: q_start = time.time()

qpu_result = dnx.min_vertex_cover(vcv3, qpu_sampler)

q_end = time.time()

qpu_result_time = (q_end - q_start)

print("3. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)

3. PROBLEM - QPU ÇÖZÜM SÜRESİ: 7.12906289100647
```

Şekil 6.1.8: Minimum Vertex Cover, 3. Problemin QPU ile Çözümlenmesi

Şekil 6.1.8'de, Minimum Vertex Cover problemine örnek olacak şekilde yarattığımız son problem olan Üçüncü Problemin QPU ile çözümlenmesini gözlemliyoruz. Çözüm süreci ~7.13 saniyede gerçekleşmiş. Önceki problemlerin çözüm süreleri ile kıyaslandığında ~1.53 saniyelik artık varmış gibi gözükebilir. Bu artışın HTTP istek/yanıt gecikmelerindeki değişkenlikten kaynaklı olması daha muhtemel gözükmektedir.

Üçüncü Problem - CPU ile Çözümleme

```
In [38]: c_start = time.time()

cpu_result = dnx.min_vertex_cover(vcv3, cpu_sampler)

c_end = time.time()

cpu_result_time = (c_end - c_start)

print("3. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)
```

3. PROBLEM - CPU ÇÖZÜM SÜRESİ: 220.4924087524414

Şekil 6.1.9. Minimum Vertex Cover, 3. Problemin CPU ile Çözümlenmesi

Şekil 6.1.9'da Üçüncü Probleme klasik CPU ile çözüm getirilmesinin ~220.49 saniye süregünü gözlemliyoruz. Önceki problemlerin çözümlerinin ~0.004 ve ~3.04 saniyede gerçekleştiği düşünüldüğünde, problemdeki birim sayısı sadece 5 artmasına rağmen çözüm süresinde çok ciddi bir artış olduğu söylenebilir.

Problemin klasik CPU'lar üzerindeki fizibilitesi düşünüldüğünde, bu noktadan sonra işlem sürelerinin gittikçe katlanarak artacağını söyleyebiliriz. Oysa kuantum işlem birimleri, bu tür yavaşlamalar olmadan büyük hacimli problemlere de mantıklı sürelerde çözüm getirebilme potansiyeli barındırmaktadır.

Birinci Problem - Çözümlerin Karşılaştırılması

```
In [39]: print("1. PROBLEM - QPU SONUÇ: ", qpu_result_a)
print("1. PROBLEM - CPU SONUÇ: ", cpu_result_a)

1. PROBLEM - QPU SONUÇ:  [1, 2, 3, 4, 5, 7, 8, 9]
1. PROBLEM - CPU SONUÇ:  [0, 1, 2, 3, 4, 5, 6, 7, 9]
```

İkinci Problem - Çözümlerin Karşılaştırılması

```
In [40]: print("2. PROBLEM - QPU SONUÇ: ", qpu_result_b)
print("2. PROBLEM - CPU SONUÇ: ", cpu_result_b)

2. PROBLEM - QPU SONUÇ:  [0, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 15, 16]
2. PROBLEM - CPU SONUÇ:  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 1
4, 15, 16, 17, 19]
```

Üçüncü Problem - Çözümlerin Karşılaştırılması

```
In [41]: print("3. PROBLEM - QPU SONUÇ: ", qpu_result)
print("3. PROBLEM - CPU SONUÇ: ", cpu_result)

3. PROBLEM - QPU SONUÇ:  [2, 3, 4, 5, 6, 8, 9, 12, 14, 15, 16, 17, 19, 2
1, 24]
3. PROBLEM - CPU SONUÇ:  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 1
4, 15, 16, 17, 18, 19, 20, 21, 22, 24]
```

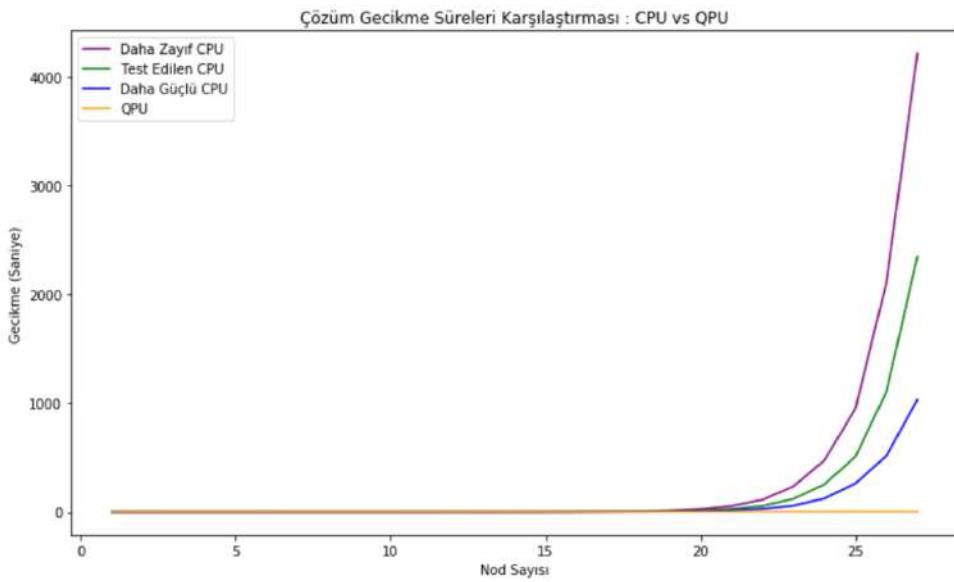
Şekil 6.1.10. Minimum Vertex Cover, Çözümlerin Karşılaştırılması

Şekil 6.1.10'da problemlere QPU ve CPU ile verilen çözümlemeler karşılaştırılmaktadır. Minimum Vertex Cover problemi özelinde düşünüldüğünde çözümler birbirine yakın olsa da, QPU çözümleri için tam anlamıyla tutarlı diyemeyiz. Bunun en temel sebeplerinden biri kuantum bilgisayarların yapısal olarak çeşitli rastgelelik durumlarından etkilenebilmesi ve yüksek gürültü oranlarıdır. Bu tür faktörlerin çözümün tutarlılığı üzerindeki etkisinin azaltılması amacıyla, çözümlere yönelik ölçümlemeler genellikle bu çalışmada gibi bir adet çözümlemenin aksine üst üste birçok kez çözümleme sağlanarak gerçekleştirilmektedir.

N (nod-sayısı)	CPU gecikmesi (saniye)	QPU gecikmesi (saniye)
1	0.0	2.37
2	0.49	1.92
3	0.0	1.90
4	0.0	1.90
5	0.0	1.91
6	0.1	1.91
7	0.0	1.92
8	0.0	1.95
9	0.0	1.95
10	0.2	2.00
11	0.3	2.12
12	0.64	2.06
13	0.15	2.04
14	0.26	2.08
15	0.37	2.21
16	0.83	2.19
17	1.91	2.19
18	3.37	2.40
19	7.26	2.53
20	13.8	2.42
21	28.6	2.86
22	56.1	2.76
23	120.4	2.32
24	253.5	2.12
25	508.2	2.82
26	1103.7	2.87
27	2343.2	1.92

Şekil 6.1.11. QPU ve CPU Ortalama İşlem Gecikmesi Tablosu

Şekil 6.1.11'de Minimum Vertex Cover probleminin sahip olduğu birim sayısı ile QPU ve CPU gecikmeleri arasında nasıl bir gecikme olduğu gösterilmeye çalışılmıştır. QPU ünitelerinin belirli bir karlılık noktasına kadar klasik CPU'lardan daha yavaş olduğu, ancak 18 adet birimden itibaren CPU'ya kıyasla gittikçe daha hızlı hale geldikleri gözlenebilmektedir. Daha yüksek birim sayılarında, örneğin 27 adet birim barındıran problemlerde ise, CPU ~2343 saniyede çözüme ulaşırken kuantum işlem biriminin çözüm hızında neredeyse hiçbir değişiklik olmamıştır.



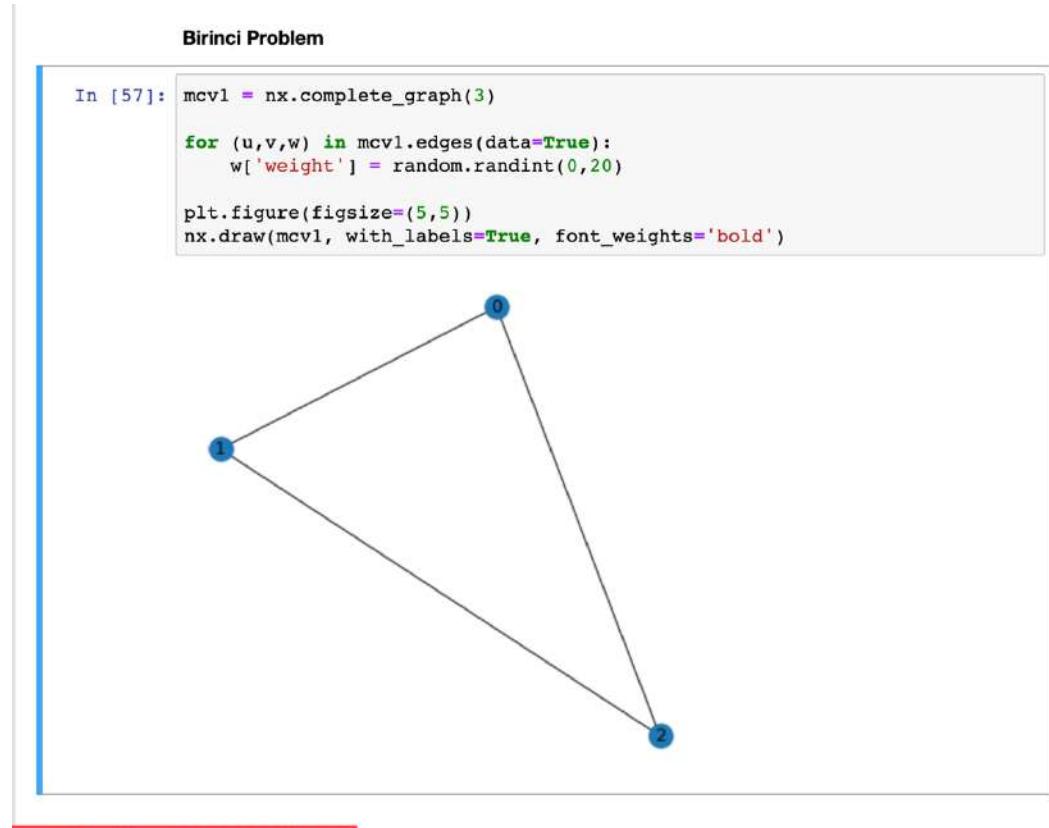
Şekil 6.1.12. QPU ve CPU İşlem Gecikmesi Grafiği

Şekil 6.1.11'de belirtilen işlem gecikmesi tablosunun daha iyi anlaşılabilmesi için Şekil 6.1.12'de gösterilen grafiği oluşturduk. Bu grafikte anlaşılması gereken en önemli noktalardan biri, CPU'ların işlem çözümlerini gerçekleştirmeye konusundaki doğasıdır. Çalışmamızda kullandığımız CPU, yeşil renk ile belirtilen hatta görülebilecek 18. birimden itibaren gittikçe daha yavaş performans göstermektedir ve bir yerden sonra çözüm süresi üssel olarak artmakta ve fizibilitesini yitirmektedir. Bu durumda daha az işlem gücüne sahip bir CPU, QPU ile kıyaslandığında sahip olduğu hız avantajını daha erken bir noktada kaybederken, daha yüksek işlem gücüne sahip bir CPU, hız avantajını daha ileride bir noktada kaybetmektedir. Ancak avantaj kaybedilen noktadan itibaren gerçekleşen üssel artış tüm CPU'lar için ortaktır. Bu artış, CPU'ların çözüm getirebileceği maksimum problem boyutunu sınırlamaktadır. Oysa kuantum bilgisayarların bu problemleri ideal durumda gerçekleştirebildiği düşünüldüğünde, problem boyutu büyündüğünde çözümlerin sürelerindeki artış üssel olarak değil, lineer olarak gerçekleşecektir.

6.2 Map Coloring Problemi

Bu bölümde Map Coloring problemine ilişkin örnek olacak üç farklı problemin çözümlenmesi üzerine çaba gösterdik. Bu problemleri de D-Wave QPU ve klasik CPU kullanarak çözümledik ve performanslarını raporladık. Son olarak elde edilen sonuçları karşılaştırdık ve yorumladık.

Map Coloring algoritmasının amacı harita örneği üzerinden aşağıdaki gibi özetlenebilmektedir. Bir ülkeyi ve şehirlerini içeren bir harita düşünelim. Bu şehirlerin her birini bir renge boyamak istiyoruz fakat bu boyama öyle bir şekilde gerçekleşmelidir ki birbirine herhangi bir şekilde komşu olan iki bölge aynı renk olmamalıdır. Map Coloring problemi NP-Complete karmaşıklık sınıfına aittir.



Şekil 6.2.1. Map Coloring, 1. Problem

Şekil 6.2.1'de Map Coloring Problemine ilişkin çözümlemek amacıyla tasarladığımız ilk çizgeyi gözlemliyoruz. Bu problemde yer alan toplam birim sayısı üçtür ve problemde yer alan üç noktadan her biri birbirine bağlı durumdadır.

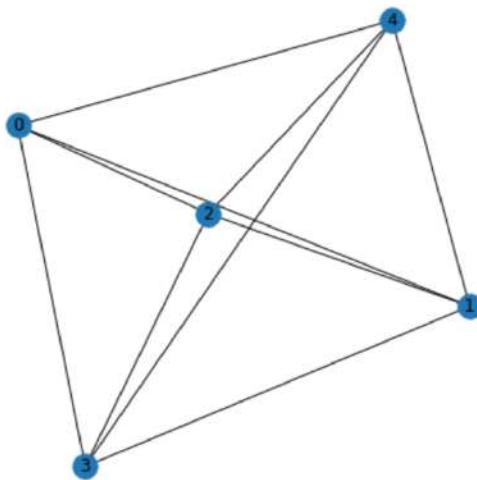
Bu durum problemi oluşturan çizgenin bir tam çizge olarak ifade edilmesini sağlamaktadır. Buna ek olarak çizgenin sahip olduğu tüm bağlantıların ağırlıkları 0 ve 20 arasında rastgele olarak belirlenmektedir.

İkinci Problem

```
In [58]: mcv2 = nx.complete_graph(5)

for (u,v,w) in mcv2.edges(data=True):
    w[ 'weight' ] = random.randint(0,20)

plt.figure(figsize=(5,5))
nx.draw(mcv2, with_labels=True, font_weights='bold')
```



Şekil 6.2.2. Map Coloring, 2. Problem

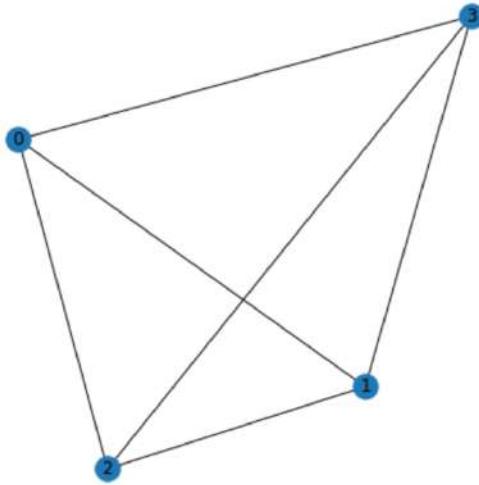
Şekil 6.2.2'de Map Coloring problemi için oluşturduğumuz ikinci problemi gözlemliyoruz. Bu problemde beş adet birim bulunmaktadır ve bu birimlerin arasında onları birbirine bağlayan toplamda 10 adet bağlantı yer alır. Bağlantı noktalarının ağırlıkları 0 ile 20 arasında rastgele olarak belirlenmiştir.

Üçüncü Problem

```
In [76]: mcv3 = nx.complete_graph(4)

for (u,v,w) in mcv3.edges(data=True):
    w['weight'] = random.randint(0,20)

plt.figure(figsize=(5,5))
nx.draw(mcv3, with_labels=True, font_weights='bold')
```



Şekil 6.2.3. Map Coloring, 3. Problem

Şekil 6.2.3'de Map Coloring problemi için tanımladığımız üçüncü çizge örneğini gözlemlemekteyiz. Bu çizge dört adet birimden ve bu birimleri birbirine bağlayan altı adet bağlantı noktasından meydana gelmektedir. Bağlantı noktalarının ağırlıkları 0 ve 20 arasında rastgele olarak belirlenmektedir.

Birinci Problem - QPU ile Çözümleme

```
In [64]: c_start = time.time()

qpu_result_a = dnx.min_vertex_color(mcv1, qpu_sampler)

c_end = time.time()

qpu_result_time = (c_end - c_start)
|
print("1. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)

1. PROBLEM - QPU ÇÖZÜM SÜRESİ:  5.169461011886597
```

Şekil 6.2.4. Map Coloring, 1. Problemin QPU ile Çözümlenmesi

Şekil 6.2.4'de görülebileceği gibi birinci problemin çözümü D-Wave Kuantum İşlem Birimi tarafından ~5.17 saniyede gerçekleştirılmıştır.

Birinci Problem - CPU ile Çözümleme

```
In [65]: c_start = time.time()
cpu_result_a = dnx.min_vertex_color(mcv1, cpu_sampler)
c_end = time.time()
cpu_result_time = (c_end - c_start)
print("1. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)

1. PROBLEM - CPU ÇÖZÜM SÜRESİ:  0.0037550926208496094
```

Şekil 6.2.5. Map Coloring, 1. Problemin CPU ile Çözümlenmesi

Şekil 6.2.5'de ise aynı problemin klasik CPU ile çözümlenmesinin ~0.003 saniye süredüğünü gözlemliyoruz. Bu problem özelinde düşünüldüğünde üç adet birimin bulunduğu bir tam çizgede CPU'ların D-Wave QPU'ya kıyasla daha hızlı çalıştığını söylemek mümkündür. QPU işlem gecikmesi konusunda esas payı yine HTTP istek/yanıt gecikmeleri almaktadır. İlerleyen problemlerde birim sayısı arttırıldığında QPU ve CPU işlem sürelerinin nasıl değiştiğini inceleyeceğiz.

İkinci Problem - QPU ile Çözümleme

```
In [66]: c_start = time.time()
qpu_result_b = dnx.min_vertex_color(mcv2, qpu_sampler)
c_end = time.time()
qpu_result_time = (c_end - c_start)
print("2. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)

2. PROBLEM - QPU ÇÖZÜM SÜRESİ:  4.943257808685303
```

Şekil 6.2.6. Map Coloring, 2. Problemin QPU ile Çözümlenmesi

Şekil 6.2.6'da Map Coloring problemi için oluşturulan ikinci örnek çizgenin QPU tarafından çözümlenmesinin ~4.94 saniye süredüğünü gözlemlemekteyiz. İkinci problem, birinci problemden 2 adet daha fazla birime sahiptir. Buna rağmen QPU işlem biriminin önceki probleme göre daha kısa sürede işlemi bitirdiğini gözlemliyoruz. Bu durumu daha önce de bahsedildiği gibi HTTP istek/yanıt gecikmelerine bağlamak mümkündür.,

İkinci Problem - CPU ile Çözümleme

```
In [67]: c_start = time.time()

cpu_result_b = dnx.min_vertex_color(mcv2, cpu_sampler)

c_end = time.time()

cpu_result_time = (c_end - c_start)

print("2. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)

2. PROBLEM - CPU ÇÖZÜM SÜRESİ:  110.88750195503235
```

Şekil 6.2.7. Map Coloring, 2. Problemin CPU ile Çözümlenmesi

Şekil 6.2.7'de ikinci örnek problemin CPU tarafından çözümlenmesinin ~110.89 saniye süregünü gözlemleyebiliriz. Bu noktada, CPU'nun işlemi gerçekleştirme hızının önceki probleme kıyasla ciddi oranda düştüğü ve QPU'nun performansının altında kaldığını söyleyebiliriz.

Üçüncü Problem - QPU ile Çözümleme

```
In [77]: c_start = time.time()

qpu_result = dnx.min_vertex_color(mcv3, qpu_sampler)

c_end = time.time()

qpu_result_time = (c_end - c_start)

print("3. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)

3. PROBLEM - QPU ÇÖZÜM SÜRESİ:  5.323257923126221
```

Şekil 6.2.8. Map Coloring, 3. Problemin QPU ile Çözümlenmesi

Şekil 6.2.8'de dört adet birimden oluşan üçüncü çizgenin QPU tarafından çözümlenmesinin ~5.32 saniye süregünü inceleyebiliriz. Bu süre, diğer problemlerde olduğu gibi QPU işlem biriminin ortalama yanıt süresi dolaylarındadır ve en azından mevcut problem boyutunda herhangi bir işlemsel yavaşlama saptanmadığı düşünülmektedir.

Üçüncü Problem - CPU ile Çözümleme

```
In [78]: c_start = time.time()

cpu_result = dnx.min_vertex_color(mcv3, cpu_sampler)

c_end = time.time()

cpu_result_time = |(c_end - c_start)

print("3. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)

3. PROBLEM - CPU ÇÖZÜM SÜRESİ:  0.11782503128051758
```

Şekil 6.2.9. Map Coloring, 3. Problemin CPU ile Çözümlenmesi

Şekil 6.2.9'da aynı problemin CPU ile çözümlenmesinin ~0.12 saniyede gerçekleştiğini gözlemlemekteyiz. Bu çözüm, üç birimden oluşan birinci problemin çözümü ile kıyaslandığında daha yavaş, ancak beş birimden oluşan ikinci problemin çözümü ile kıyaslandığında çok daha hızlıdır. Çalışmamızda gerçekleştirdiğimiz bu deneye göre klasik CPU'larda, ufak problem hacimlerinde bile ciddi performans yavaşlamaları görülebilmektedir.

Birinci Problem - Çözümlerin Karşılaştırılması

```
In [79]: print("1. PROBLEM - QPU SONUÇ: ", qpu_result_a)
print("1. PROBLEM - CPU SONUÇ: ", cpu_result_a)

1. PROBLEM - QPU SONUÇ: {0: 0, 1: 1, 2: 2}
1. PROBLEM - CPU SONUÇ: {0: 2, 1: 0, 2: 1}
```

İkinci Problem - Çözümlerin Karşılaştırılması

```
In [603]: print("2. PROBLEM - QPU SONUÇ: ", qpu_result_b)
print("2. PROBLEM - CPU SONUÇ: ", cpu_result_b)

2. PROBLEM - QPU SONUÇ: {0: 2, 1: 3, 2: 0, 3: 1, 4: 4}
2. PROBLEM - CPU SONUÇ: {0: 3, 1: 4, 2: 1, 3: 2, 4: 0}
```

Üçüncü Problem - Çözümlerin Karşılaştırılması

```
In [81]: print("3. PROBLEM - QPU SONUÇ: ", qpu_result)
print("3. PROBLEM - CPU SONUÇ: ", cpu_result)

1. PROBLEM - QPU SONUÇ: {0: 2, 2: 0, 3: 2}
1. PROBLEM - CPU SONUÇ: {0: 1, 1: 2, 2: 0, 3: 3}
```

Şekil 6.2.10. Map Coloring, QPU ve CPU Çözümlerinin Karşılaştırılması

Şekil 6.2.10'da gözlenebileceği üzere, QPU ve CPU'nun ürettiği çözümler arasında tutarsızlıklar olabilmektedir. Bu tutarsızlıkların sebebi çoklu çözüm gerçekleştirilmeyen QPU kullanımlarına ek olarak, kimi zaman problemin birden fazla olası çözüme de sahip olabilmesidir. Daha ileri bir araştırmada, çoklu çözüm kullanılarak daha tutarlı cevaplar edinilmesi mümkün olabilir.

6.3 Maximum Clique Problemi

Bu bölümde D-Wave QPU ve klasik CPU çözümleyiciler kullanarak Network-X kütüphanesi ile oluşturduğumuz örnek üç adet Maximum Clique problemini çözümledik. Çözümlerin sonuçlarını QPU ve CPU için ayrı ayrı raporlandırdık ve çözümlerin karşılaştırılmasını sağladık.

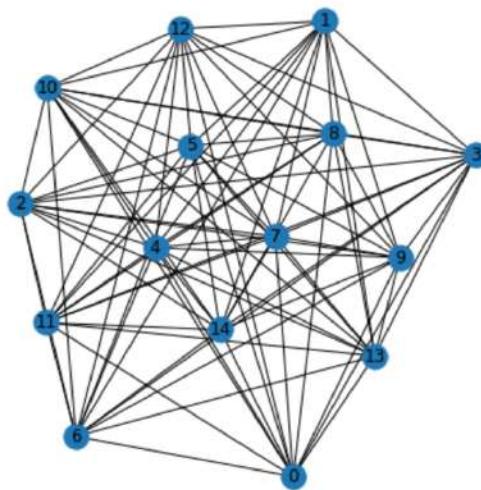
Maximum Clique, bir çizgede olabilmesi mümkün olan en geniş birim hacmine sahip Clique'i bulmayı amaçlamaktadır. NP-Hard karmaşıklık sınıfına ait bir problemdir.

Birinci Problem

```
In [121]: mxcl = nx.turan_graph(15, 5)

for (u,v,w) in mxcl.edges(data=True):
    w[ 'weight' ] = random.randint(0,20)

plt.figure(figsize=(5,5))
nx.draw(mxcl, with_labels=True, font_weights='bold')
```



Şekil 6.3.1. Maximum Clique, 1. Problem

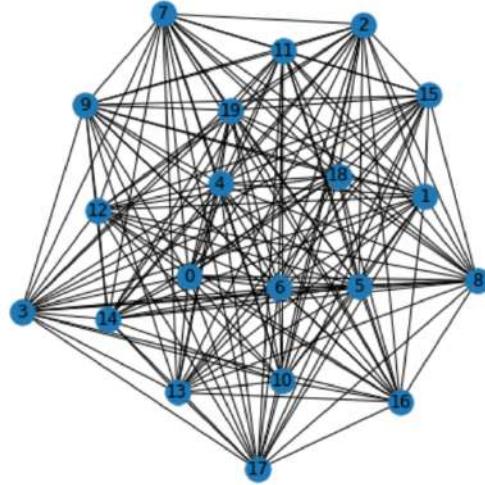
Şekil 6.3.1'de görüleceği üzere, Maximum Clique probleminin çözümlenmesi için tanımlanan ilk çizge, önceki problemlerde kullanılan tam çizgelerin aksine Turan tipi bir çizgedir. Bu çizgedeki bağlantıların arasındaki ağırlıklar belirlenirken de 0 ve 20 arasında rastgele değerler belirlenmiştir.

İkinci Problem

```
In [125]: mxc2 = nx.turan_graph(20, 8)

for (u,v,w) in mxc2.edges(data=True):
    w['weight'] = random.randint(0,20)

plt.figure(figsize=(5,5))
nx.draw(mxc2, with_labels=True, font_weights='bold')
```



Şekil 6.3.2. Maximum Clique, 2. Problem

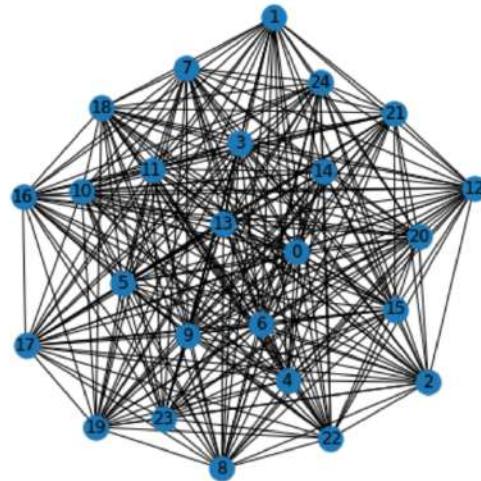
Şekil 6.3.2'de belirtilen ikinci problem, birinci probleme kıyasla biraz daha karmaşık bir problem olmak ile birlikte çizgenin tanımlanmasında aynı şekilde Turan tipi çizgelerden yararlanılmıştır. Bağlantıların arasındaki ağırlıkların belirlenmesinde 0 ve 20 arasında rastgele sayıların seçilmesi esas alınmıştır.

Üçüncü Problem

```
In [126]: mxc3 = nx.turan_graph(25, 13)

for (u,v,w) in mxc3.edges(data=True):
    w['weight'] = random.randint(0,20)

plt.figure(figsize=(5,5))
nx.draw(mxc3, with_labels=True, font_weights='bold')
```



Şekil 6.3.3. Maximum Clique, 3. Problem

Şekil 6.3.3'de ise Maximum Clique problemine örnek olarak oluşturulmuş üçüncü çizgeyi gözlemlileyebiliriz. Bu çizge birinci ve ikinci çizgelere göre biraz daha karmaşıktır ve 25 birimden meydana gelmektedir. Turan formunda çizge kullanılmıştır. Bağlantı ağırlıklarının belirlenmesinde 0 ve 20 arasında rastgele değerler seçilmiştir.

Birinci Problem - QPU ile Çözümleme

```
In [604]: c_start = time.time()

qpu_result_a = dnx.maximum_clique(mxc1, qpu_sampler)

c_end = time.time()

qpu_result_time = (c_end - c_start)

print("1. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)

1. PROBLEM - QPU ÇÖZÜM SÜRESİ:  4.724148988723755
```

Şekil 6.3.4. Maximum Clique, 1. Problemin QPU ile Çözümlenmesi

Şekil 6.3.4'de birinci problemin QPU tarafından çözümlenmesinin ~4.72 saniye sürdüğünü gözlemeğekteyiz.

Birinci Problem - CPU ile Çözümleme

```
In [605]: c_start = time.time()
|
cpu_result_a = dnx.maximum_clique(mxc1, cpu_sampler)

c_end = time.time()

cpu_result_time = (c_end - c_start)

print("1. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)
1. PROBLEM - CPU ÇÖZÜM SÜRESİ: 0.06234598159790039
```

Şekil 6.3.5. Maximum Clique, 1. Problemin CPU ile Çözümlenmesi

Şekil 6.3.5'den anlaşılabileceği üzere birinci problemin CPU tarafından çözümlenmesi ise ~0.06 saniye sürmektedir. Buna dayalı olarak problemin sahip olduğu boyutlarda CPU'nun QPU'ya kıyasla daha hızlı performans gösterdiğini söylemek mümkündür.

İkinci Problem - QPU ile Çözümleme

```
In [606]: c_start = time.time()

qpu_result_b = dnx.maximum_clique(mxc2, qpu_sampler)

c_end = time.time()

qpu_result_time = (c_end - c_start)

print("2. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)
2. PROBLEM - QPU ÇÖZÜM SÜRESİ: 4.968411922454834
```

Şekil 6.3.6. Maximum Clique, 2. Problemin QPU ile Çözümlenmesi

Şekil 6.3.6'da ikinci problemin QPU tarafından çözümlenmesinin yaklaşık ~4.97 saniye sürdüğünü görüyoruz. Önceki problemin QPU tarafından çözümlenme süresi ile kıyasladığımızda arada anlamlı bir fark kaydedemedik.

İkinci Problem - CPU ile Çözümleme

```
In [607]: c_start = time.time()

cpu_result_b = dnx.maximum_clique(mxc2, cpu_sampler)

c_end = time.time()

cpu_result_time = (c_end - c_start)

print("2. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)

2. PROBLEM - CPU ÇÖZÜM SÜRESİ:  1.8269367218017578
```

Şekil 6.3.7. Maximum Clique, 2. Problemin CPU ile Çözümlenmesi

Şekil 6.3.7'de ikinci problemin CPU tarafından çözümlenmesinin ise ~1.83 saniye süredüğünü görüyoruz. Bu süre, birinci problemin çözüm süresiyle kıyaslandığında bariz düzeyde daha fazla olmasına karşın, hala CPU'nun performansı, QPU'ya kıyasla daha yüksektir.

Üçüncü Problem - QPU ile Çözümleme

```
In [608]: c_start = time.time()

qpu_result = dnx.maximum_clique(mxc3, qpu_sampler)

c_end = time.time()

qpu_result_time = (c_end - c_start)

print("3. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)

3. PROBLEM - QPU ÇÖZÜM SÜRESİ:  3.9754130840301514
```

Şekil 6.3.8. Maximum Clique, 3. Problemin QPU ile Çözümlenmesi

Şekil 6.3.8'de üçüncü problemin QPU tarafından çözümlenmesinin ~3.97 saniye süredüğünü görüyoruz. Birinci ve ikinci problem ile kıyaslandığında işlem süreleri konusunda hala anlamlı bir artış olmadığını söyleyebiliriz.

Üçüncü Problem - CPU ile Çözümleme

```
In [612]: c_start = time.time()

cpu_result = dnx.maximum_clique(mxc3, cpu_sampler)

c_end = time.time()

cpu_result_time = (c_end - c_start)

print("3. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)

3. PROBLEM - CPU ÇÖZÜM SÜRESİ: 57.896332025527954
```

Şekil 6.3.9. Maximum Clique, 3. Problemin CPU ile Çözümlenmesi

Şekil 6.3.9'da görülebileceği üzere üçüncü problemde CPU'nun işlem süresi ciddi şekilde artarak ~57.89 saniyeye ulaşmıştır. Bu durumda QPU'nun artık CPU'ya kıyasla daha hızlı performans sağladığını söylemek doğru olur.

Birinci Problem - Çözümlerin Karşılaştırılması

```
In [609]: print("1. PROBLEM - QPU SONUÇ: ", qpu_result_a)
print("1. PROBLEM - CPU SONUÇ: ", cpu_result_a)

1. PROBLEM - QPU SONUÇ: [1, 5, 8, 10, 14]
1. PROBLEM - CPU SONUÇ: [2, 3, 7, 9, 12]
```

İkinci Problem - Çözümlerin Karşılaştırılması

```
In [610]: print("2. PROBLEM - QPU SONUÇ: ", qpu_result_b)
print("2. PROBLEM - CPU SONUÇ: ", cpu_result_b)

2. PROBLEM - QPU SONUÇ: [0, 2, 4, 6, 10, 12, 14, 18]
2. PROBLEM - CPU SONUÇ: [1, 3, 4, 7, 10, 11, 16, 18]
```

Üçüncü Problem - Çözümlerin Karşılaştırılması

```
In [613]: print("3. PROBLEM - QPU SONUÇ: ", qpu_result)
print("3. PROBLEM - CPU SONUÇ: ", cpu_result)

3. PROBLEM - QPU SONUÇ: [0, 2, 3, 5, 8, 9, 11, 13, 16, 18, 20, 22, 23]
3. PROBLEM - CPU SONUÇ: [0, 2, 4, 5, 7, 10, 11, 13, 16, 17, 20, 22, 23]
```

Şekil 6.3.10. Maximum Clique, QPU ve CPU Çözümlerinin Karşılaştırılması

Şekil 6.3.10 incelendiğinde, önceki problemlerde de benzeri görüldüğü şekilde; QPU'nun hesapladığı sonuçların tutarlılığı üzerinde şüpheler bulunmaktadır. Sonuçların tutarlılığının arttırılması için çoklu hesaplama gerçekleştirilmesi önem arz etmektedir. Yine de sonuçlar incelendiğinde belirli bir düzeyde istenen sonuçlara yakınsadığı fark edilmektedir.

6.4 Minimum Maximal Matching Problemi

Bu bölümde Minimum Maximal Matching problemine uyacak şekilde ürettiğimiz üç adet çizgeye, D-Wave QPU ve klasik CPU çözümleyiciler kullanarak çözümlemeler ürettiğimiz. Elde ettiğimiz sonuçları ve işlem sürelerini raporlandırdık ve çalışmamızda belirttik.

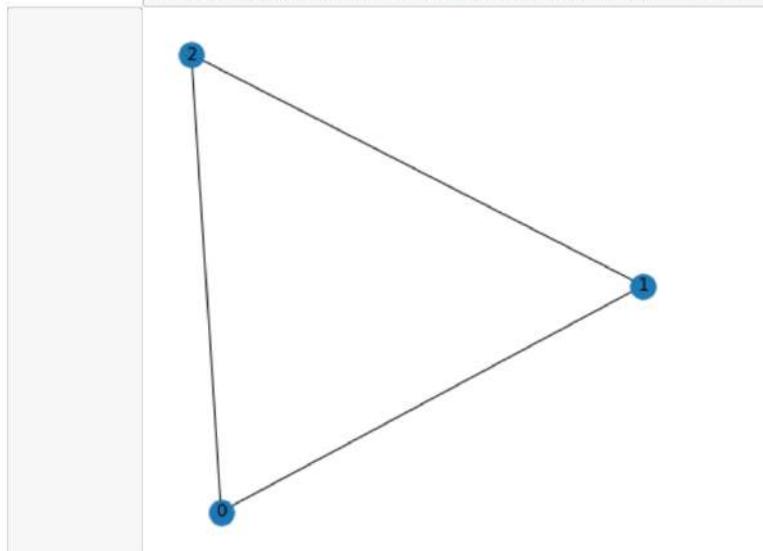
Matching, çizgelerde komşu olmayan kenarlardan oluşan alt kümeleri ifade etmektedir. Maximum matching, çizge dahilinde Matching kurallarına uyacak biçimde mümkün olan en fazla sayıda kenarın varlığını ifade eden terimdir. Minimum maximal matching ise, kenar sayısını minimumda tutacak şekilde çizgede mümkün olan maksimum Matching'e ulaşmayı hedefleyen problemdir. NP-Hard karmaşıklık sınıfına dahil bir problemdir.

Birinci Problem

```
In [156]: mmm1 = nx.complete_graph(3)

for (u,v,w) in mmm1.edges(data=True):
    w['weight'] = random.randint(0,20)

plt.figure(figsize=(5,5))
nx.draw(mmm1, with_labels=True, font_weights='bold')
```



Şekil 6.4.1. Minimum Maximal Matching, 1. Problem

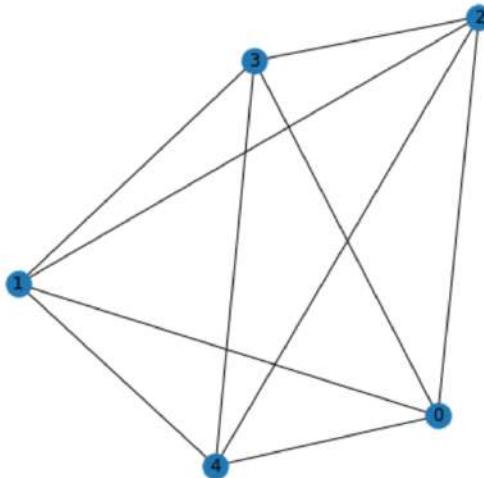
Şekil 6.4.1'de Minimum Maximal Matching problemi için oluşturduğumuz ilk örnek problemi gözlemleyebiliriz. Bu çizge üç adet birimden ve bu birimleri birbirine bağlayan üç adet bağlantından oluşmaktadır. Birimleri bağlayan kenarların ağırlıkları 0 ve 20 arasında rastgele olacak şekilde belirlenmiştir.

İkinci Problem

```
In [157]: mmm2 = nx.complete_graph(5)

for (u,v,w) in mmm2.edges(data=True):
    w[ 'weight' ] = random.randint(0,20)

plt.figure(figsize=(5,5))
nx.draw(mmm2, with_labels=True, font_weights='bold')
```



Şekil 6.4.2. Minimum Maximal Matching, 2. Problem

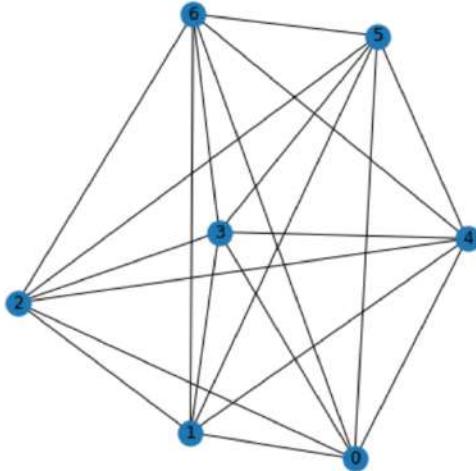
Şekil 6.4.2'de oluşturduğumuz ikinci problemi inceleyebiliriz. Bu problem beş adet birim ve bu birimleri birbirine bağlayan on adet bağlantından meydana gelmektedir. Bu bağlantıların ağırlıkları 0 ve 20 arasında rastgele olarak belirlenmiştir.

Üçüncü Problem

```
In [158]: mmm3 = nx.complete_graph(7)

for (u,v,w) in mmm3.edges(data=True):
    w['weight'] = random.randint(0,20)

plt.figure(figsize=(5,5))
nx.draw(mmm3, with_labels=True, font_weights='bold')
```



Şekil 6.4.3. Minimum Maximal Matching, 3. Problem

Şekil 6.4.3'de Minimum Maximal Matching problemi için oluşturduğumuz son problem örneğini inceleyebiliriz. Bu örnek yedi adet birim ve bu birimleri birbirine bağlayan yirmi bir adet bağlantıdan meydana gelmektedir. Bu bağlantıların ağırlıkları da diğer problemlerde olduğu gibi 0 ve 20 arasında rastgele şekilde belirlenmektedir.

Birinci Problem - QPU ile Çözümleme

```
In [159]: c_start = time.time()

qpu_result_a = dnx.min_maximal_matching(mmm1, qpu_sampler)

c_end = time.time()

qpu_result_time = (c_end - c_start)

print("1. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)

1. PROBLEM - QPU ÇÖZÜM SÜRESİ:  5.174998998641968
```

Şekil 6.4.4. Minimum Maximal Matching, 1. Problemin QPU ile Çözümlenmesi

Şekil 6.4.4'de görülebilecek üzere, birinci problemin QPU kullanılarak çözümlenmesi ~5.17 saniyede gerçekleşmiştir.

Birinci Problem - CPU ile Çözümleme

```
In [160]: c_start = time.time()

cpu_result_a = dnx.min_maximal_matching(mmm1, cpu_sampler)

c_end = time.time()

cpu_result_time = (c_end - c_start)

print("1. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)

1. PROBLEM - CPU ÇÖZÜM SÜRESİ: 0.0010380744934082031
```

Şekil 6.4.5. Minimum Maximal Matching, 1. Problemin CPU ile Çözümlenmesi

Birinci problemin CPU ile çözümlenmesi ise ~0.001 saniyede gerçekleşmiştir. QPU ile kıyaslandığında CPU, bu problem boyutunda daha hızlı performans göstermiştir. (Şekil 6.4.5) Diğer problemlerde problemin boyutunu büyütceğ ve CPU işlem hızının yavaşlayıp yavaşlamayacağını test edeceğiz.

İkinci Problem - QPU ile Çözümleme

```
In [161]: c_start = time.time()

qpu_result_b = dnx.min_maximal_matching(mmm2, qpu_sampler)

c_end = time.time()

qpu_result_time = (c_end - c_start)

print("2. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)

2. PROBLEM - QPU ÇÖZÜM SÜRESİ: 6.901879787445068
```

Şekil 6.4.6. Minimum Maximal Matching, 2. Problemin QPU ile Çözümlenmesi

Şekil 6.4.6'da ikinci problemin QPU tarafından çözümlenmesinin ~6.9 saniye süրdüğünü göstermiştir. Birinci problem ile kıyaslandığında aradaki yaklaşık 1 saniyelik farkın, QPU'nun bu çeşit problemler üzerinde gösterdiği ortalama performans dikkate alındığında normal olduğu düşünülmüştür.

İkinci Problem - CPU ile Çözümleme

```
In [162]: c_start = time.time()
cpu_result_b = dnx.min_maximal_matching(mmm2, cpu_sampler)
c_end = time.time()
cpu_result_time = (c_end - c_start)
print("2. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)

2. PROBLEM - CPU ÇÖZÜM SÜRESİ: 0.006217002868652344
```

Şekil 6.4.7. Minimum Maximal Matching, 2. Problemin CPU ile Çözümlenmesi

İkinci problem, CPU tarafından ~0.006 saniyede çözümlenmiştir. Bu durumda CPU, QPU'nun sağladığı süreye kıyasla hala anlamlı ölçüde daha hızlıdır. (Şekil 6.4.7)

Üçüncü Problem - QPU ile Çözümleme

```
In [163]: c_start = time.time()
qpu_result = dnx.min_maximal_matching(mmm3, qpu_sampler)
c_end = time.time()
qpu_result_time = (c_end - c_start)
print("3. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)

3. PROBLEM - QPU ÇÖZÜM SÜRESİ: 5.537613153457642
```

Şekil 6.4.8. Minimum Maximal Matching, 3. Problemin QPU ile Çözümlenmesi

Şekil 6.4.8'de üçüncü problemin QPU tarafından çözümlenmesi için harcanan sürenin ~5.37 saniye olduğunu gözlemliyoruz. Bu durumda birinci ve ikinci problemin çözüm süreleri ile kıyaslandığında QPU performansında kayda değer bir yavaşlama saptanmamıştır.

Üçüncü Problem - CPU ile Çözümleme

```
In [164]: c_start = time.time()
cpu_result = dnx.min_maximal_matching(mmm3, cpu_sampler)
c_end = time.time()
cpu_result_time = (c_end - c_start)
print("3. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)

3. PROBLEM - CPU ÇÖZÜM SÜRESİ:  9.340500116348267
```

Şekil 6.4.9. Minimum Maximal Matching, 3. Problemin CPU ile Çözümlenmesi

Şekil 6.4.9'da, üçüncü ve son problemin CPU ile çözümlenmesinin ~9.34 saniye süregünü gözlemliyoruz. Önceki iki problem düşünüldüğünde çözüm süresinin anlamlı ölçüde arttığını söylemek doğru olacaktır. Bu noktada QPU'nun çözüm gerçekleştirdiği sürenin daha kısa olduğu düşünüldüğünde CPU'ya kıyasla da avantajlı hale geçtiği yorumunda bulunmak doğru olacaktır.

Birinci Problem - Çözümlerin Karşılaştırılması

```
In [165]: print("1. PROBLEM - QPU SONUÇ: ", qpu_result_a)
print("1. PROBLEM - CPU SONUÇ: ", cpu_result_a)

1. PROBLEM - QPU SONUÇ:  {(0, 2)}
1. PROBLEM - CPU SONUÇ:  {(0, 1)}
```

İkinci Problem - Çözümlerin Karşılaştırılması

```
In [166]: print("2. PROBLEM - QPU SONUÇ: ", qpu_result_b)
print("2. PROBLEM - CPU SONUÇ: ", cpu_result_b)

2. PROBLEM - QPU SONUÇ:  {(0, 1), (1, 3), (0, 2), (2, 4)}
2. PROBLEM - CPU SONUÇ:  {(1, 2), (0, 3)}
```

Üçüncü Problem - Çözümlerin Karşılaştırılması

```
In [167]: print("3. PROBLEM - QPU SONUÇ: ", qpu_result)
print("3. PROBLEM - CPU SONUÇ: ", cpu_result)

3. PROBLEM - QPU SONUÇ:  {(1, 5), (0, 2), (1, 4), (0, 4)}
3. PROBLEM - CPU SONUÇ:  {(1, 5), (4, 6), (0, 2)}
```

Şekil 6.4.10. Minimum Maximal Matching, QPU ve CPU Ölçümlerinin Karşılaştırılması

Şekil 6.4.10'da QPU ve CPU'nun ürettiği çözümlerin karşılaştırmasını gösterdik. Bu çözümlerde de önceki bölümlerde bahsedilen tutarlılık problemlerine rastlamak mümkündür. Bu tutarlılık problemlerinin çözülmesi için tekrarlı ölçümlerin yapılması avantajlı olacaktır.

6.5 Maximum Cut Problemi

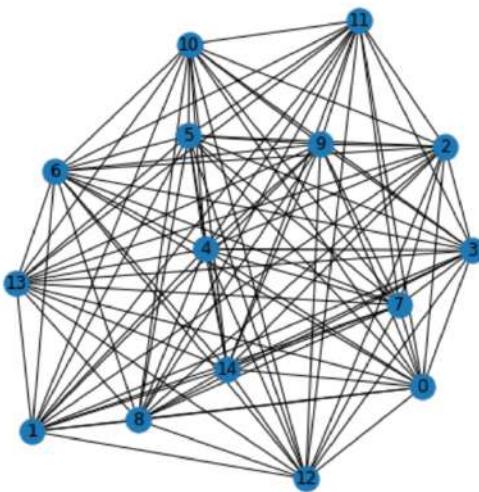
Bu bölümde Maximum Cut problemine örnek oluşturacak üç adet çizge tanımladık ve D-Wave QPU ve klasik CPU kullanarak yarattığımız örnek problemlere çözümler getirdik. QPU ve CPU'nun işlem sürelerine ek olarak elde ettiğimiz ölçüm sonuçlarını raporlandırdık.

Bir çizgede gerçekleştirilen herhangi bir kesik, mevcut diğer kesiklerin herhangi birinden daha küçük değilse o kesige maksimum kesi denilmektedir. Orijinal olarak Maximum Cut şeklinde ifade edilmektedir. NP-Complete karmaşıklık sınıfına ait bir problemdir.

```
In [194]: mxcut1 = nx.complete_graph(15)

for (u,v,w) in mxcut1.edges(data=True):
    w['weight'] = random.randint(0,20)

plt.figure(figsize=(5,5))
nx.draw(mxcut1, with_labels=True, font_weights='bold')
```



Şekil 6.5.1. Maximum Cut, 1. Problem

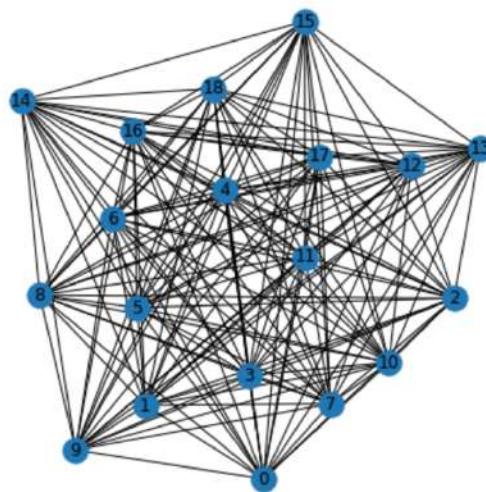
Şekil 6.5.1'de Maximum Cut problemi için örnek olarak yarattığımız birinci çizgeyi gözlemleyebiliriz. Bu çizge, on beş adet birimden ve bu birimleri birbirine

bağlayan yüz beş bağlantıdan meydana gelmektedir. Bu bağlantıların her birinin ağırlıkları 0 ve 20 arasında rastgele olacak şekilde oluşturulmuştur.

```
In [195]: mxcut2 = nx.complete_graph(19)

for (u,v,w) in mxcut2.edges(data=True):
    w['weight'] = random.randint(0,20)

plt.figure(figsize=(5,5))
nx.draw(mxcut2, with_labels=True, font_weights='bold')
```



Şekil 6.5.2. Maximum Cut, 2. Problem

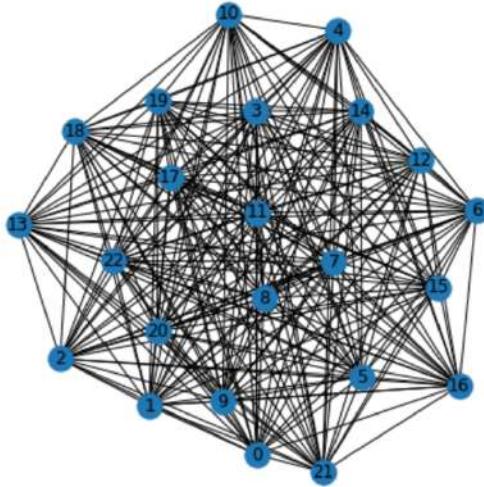
Şekil 6.5.2'de Maximum Cut problemi için oluşturduğumuz ikinci örnek çizgeyi gözlemlleyebiliriz. Bu çizge önceki probleme göre biraz daha karmaşık olup on dokuz adet birimden ve bu birimleri birbirine bağlayan yüz yetmiş bir adet bağlantıdan meydana gelmektedir. Bu bağlantıların her birinin ağırlığı 0 ve 20 arasında rastgele olacak şekilde belirlenmiştir.

Üçüncü Problem

```
In [196]: mxcut3 = nx.complete_graph(23)

for (u,v,w) in mxcut3.edges(data=True):
    w['weight'] = random.randint(0,20)

plt.figure(figsize=(5,5))
nx.draw(mxcut3, with_labels=True, font_weights='bold')
```



Şekil 6.5.3. Maximum Cut, 3. Problem

Şekil 6.5.3'de Maximum Cut problemi için örnek olarak tasarladığımız üçüncü çizgeyi inceleyebiliriz. Bu çizge birinci ve ikinci problemlerdeki çizgelere kıyasla biraz daha kompleks olup, yirmi üç adet birime ek olarak bu birimleri birbirine bağlayan iki yüz elli üç adet bağlantından oluşmaktadır. Bu bağlantıların ağırlıkları 0 ve 20 arasında rastgele olacak şekilde belirlenmiştir.

Birinci Problem - QPU ile Çözümlme

```
In [197]: c_start = time.time()

qpu_result_a = dnx.maximum_cut(mxcut1, qpu_sampler)

c_end = time.time()
|
qpu_result_time = (c_end - c_start)

print("1. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)

1. PROBLEM - QPU ÇÖZÜM SÜRESİ:  5.048703193664551
```

Şekil 6.5.4. Maximum Cut, 1. Problemin QPU ile Çözümlenmesi

Şekil 6.5.4'de görülebileceği üzere Maximum Cut problemine yönelik dizayn ettiğimiz ilk problemin QPU tarafından çözülme süresi ~5.04 saniyedir.

Birinci Problem - CPU ile Çözümleme

```
In [198]: c_start = time.time()

cpu_result_a = dnx.maximum_cut(mxcut1, cpu_sampler)

c_end = time.time()

cpu_result_time = (c_end - c_start)

print("1. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)

1. PROBLEM - CPU ÇÖZÜM SÜRESİ:  0.09636402130126953
```

Şekil 6.5.5. Maximum Cut, 1. Problemin CPU ile Çözümlenmesi

Şekil 6.5.5'de ise aynı problemin CPU ile çözülme süresini görebiliriz. Birinci problemi CPU'nun çözümlemesi ~0.09 saniye almıştır. QPU'nun peformansı ile kıyaslandığında bu problem boyutu için CPU'nun QPU'ya kıyasla daha hızlı performans gösterdiği söylenebilir. Diğer problemlerde daha büyük boyutlu çizgelerde CPU'nun performansının yavaşlayıp yavaşlamadığını test ettik.

İkinci Problem - QPU ile Çözümleme

```
In [199]: c_start = time.time()

qpu_result_b = dnx.maximum_cut(mxcut2, qpu_sampler)

c_end = time.time()

qpu_result_time = (c_end - c_start)

print("2. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)

2. PROBLEM - QPU ÇÖZÜM SÜRESİ:  7.465651988983154
```

Şekil 6.5.6. Maximum Cut, 2. Problemin QPU ile Çözümlenmesi

Şekil 6.5.6 incelendiğinde, ikinci problemin QPU tarafından çözülme süresinin ~7.46 saniye olduğu gözlemlenecektir. Bu süre önceki problemin çözülme süresi ile kıyaslandığında yaklaşık 2 saniye kadar daha uzundur. Ancak, bu süre artışının sebebinin, problemin boyutunun artışından çok HTTP istek/cevap gecikmelerinden dolayı olduğu görüşündeyiz.

İkinci Problem - CPU ile Çözümleme

```
In [200]: c_start = time.time()

cpu_result_b = dnx.maximum_cut(mxcut2, cpu_sampler)

c_end = time.time()

cpu_result_time = (c_end - c_start)
|
print("2. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)

2. PROBLEM - CPU ÇÖZÜM SÜRESİ: 1.466797113418579
```

Şekil 6.5.7: Maximum Cut, 2. Problemin CPU ile Çözümlenmesi

Şekil 6.5.7'de ikinci problemin CPU tarafından çözülmesinin ~1.46 saniye süredüğünü gözlemleyebiliriz. Bu süre her ne kadar birinci problemi çözüm süresinden daha uzun olsa da, hala QPU'ya kıyasla daha hızlıdır. Bu yüzden ikinci problemde mevcut olan problem boyutlarında CPU'nun hala QPU'ya kıyasla daha yüksek performans gösterdiğini söylemek yanlış olmayacağından emin olabiliriz.

Üçüncü Problem - QPU ile Çözümleme

```
In [201]: c_start = time.time()

qpu_result = dnx.maximum_cut(mxcut3, qpu_sampler)

c_end = time.time()

qpu_result_time = (c_end - c_start)
|
print("3. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)

3. PROBLEM - QPU ÇÖZÜM SÜRESİ: 4.111576080322266
```

Şekil 6.5.8: Maximum Cut, 3. Problemin QPU ile Çözümlenmesi

Şekil 6.5.8'de görülebileceği üzere, üçüncü problemin çözümü QPU tarafından ~4.11 saniyede gerçekleştirilmiştir. Birinci ve ikinci problemler ile kıyaslandığında artan problem boyutuna rağmen daha düşük bir sürede çözümleme gerçeklestirmesi, çözüm hesaplama süresinin önemli bir kısmının ağ kaynaklı gecikmeler olduğu düşüncemizi güçlendirmektedir.

Üçüncü Problem - CPU ile Çözümleme

```
In [202]: c_start = time.time()
cpu_result = dnx.maximum_cut(mxcut3, cpu_sampler)
c_end = time.time()
cpu_result_time = (c_end - c_start)
print("3. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)

3. PROBLEM - CPU ÇÖZÜM SÜRESİ: 46.15278220176697
```

Şekil 6.5.9. Maximum Cut, 3. Problemin CPU ile Çözümlenmesi

Şekil 6.5.9'da ise üçüncü problemin CPU tarafından çözümlenmesi ~46.15 saniyede gerçekleşmiştir. Birinci ve ikinci problemler ile kıyaslandığında ciddi bir artış olmasına ek olarak, QPU ile kıyaslandığında daha yavaş kalındığı bir noktaya ulaşıldığından mevcut problem boyutu için QPU'nun CPU'dan daha avantajlı olduğunu söyleyebiliriz.

Birinci Problem - Çözümlerin Karşılaştırılması

```
In [203]: print("1. PROBLEM - QPU SONUÇ: ", qpu_result_a)
print("1. PROBLEM - CPU SONUÇ: ", cpu_result_a)

1. PROBLEM - QPU SONUÇ: {1, 4, 6, 10, 11, 12, 14}
1. PROBLEM - CPU SONUÇ: {0, 3, 4, 5, 7, 10, 13}
```

İkinci Problem - Çözümlerin Karşılaştırılması

```
In [204]: print("2. PROBLEM - QPU SONUÇ: ", qpu_result_b)
print("2. PROBLEM - CPU SONUÇ: ", cpu_result_b)

2. PROBLEM - QPU SONUÇ: {0, 1, 4, 6, 7, 8, 14, 15}
2. PROBLEM - CPU SONUÇ: {1, 2, 3, 4, 10, 11, 13, 15, 16, 17}
```

Üçüncü Problem - Çözümlerin Karşılaştırılması

```
In [205]: print("3. PROBLEM - QPU SONUÇ: ", qpu_result)
print("3. PROBLEM - CPU SONUÇ: ", cpu_result)

3. PROBLEM - QPU SONUÇ: {0, 1, 4, 6, 7, 8, 11, 15, 17, 18, 19, 22}
3. PROBLEM - CPU SONUÇ: {0, 5, 7, 10, 12, 13, 15, 18, 19, 20, 21}
```

Şekil 6.5.10. Maximum Cut, QPU ve CPU Çözümlerinin Karşılaştırılması

Şekil 6.5.10'dan gözleyebileceğimiz üzere QPU ve CPU kaynaklı olarak üretilen çözümler birbiri ile tam anlamıyla örtüşmemektedir. Bunun sebebi önceki bölümlerde de bahsedilen gürültü, tek sefer ölçümleme, çok çözümlülük gibi faktörler olabilmektedir.

6.6 Traveling Salesperson Problemi

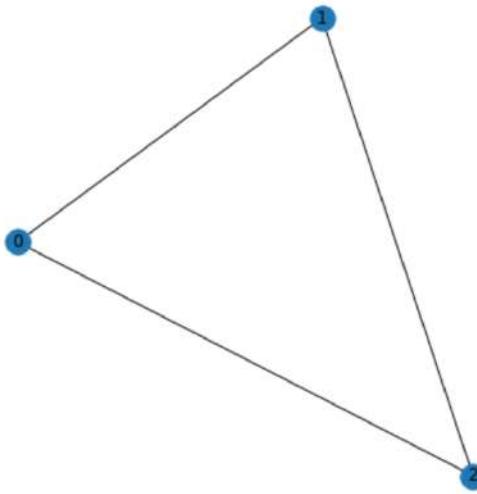
Bu bölümde Traveling Salesperson problemine örnek olacak şekilde üç farklı çizge oluşturduk. Oluşturduğumuz çizgeleri D-Wave QPU ve CPU kullanarak çözümledik ve elde ettiğimiz sonuçlara ek olarak hesaplama zamanlarını da kaydedip raporlandırdık.

Traveling salesperson problemi takip eden şekilde basitçe ifade edilebilir. Belirli sayıda birimden ve belirli sayıda bağlantıdan oluşan bir çizge düşünelim. Birimler şehirleri, bağlantılar da şehirler arası yolları temsil ediyor olsun. Bu şehirlerin tamamını gezmek isteyen bir seyyah, bunu öyle bir şekilde yapmak istiyor ki, bütün şehirleri gezmiş olmaya ek olarak, kat ettiği toplam yol mümkün olduğunda kısa olsun.

```
In [217]: tsp1 = nx.complete_graph(3)

for (u,v,w) in tsp1.edges(data=True):
    w['weight'] = random.randint(10,50)

plt.figure(figsize=(5,5))
nx.draw(tsp1, with_labels=True, font_weights='bold')
```



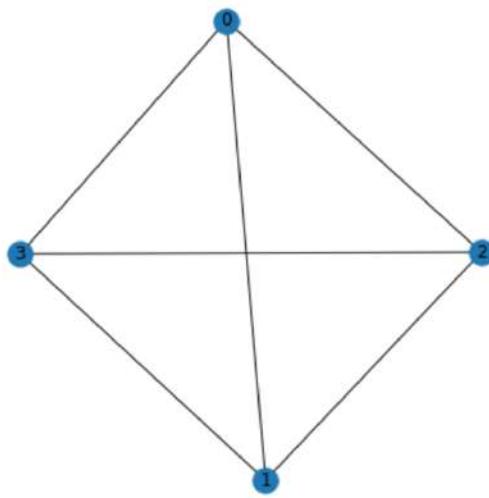
Şekil 6.6.1. Traveling Salesperson, 1. Problem

Şekil 6.6.1'de Traveling Salesperson problemi için oluşturduğumuz üç adet örnek çizgeden ilkini görebiliriz. Bu çizge üç adet birimden ve bu birimleri birbirine bağlayan üç adet bağlantıdan oluşmaktadır. Buna ek olarak, şehirler arası uzaklıklarını temsil etmek üzere bağlantıların her birine rastgele 10 ve 50 arasında ağırlıklar atanmıştır.

```
In [219]: tsp2 = nx.complete_graph(4)

for (u,v,w) in tsp2.edges(data=True):
    w['weight'] = random.randint(10,50)

plt.figure(figsize=(5,5))
nx.draw(tsp2, with_labels=True, font_weights='bold')
```



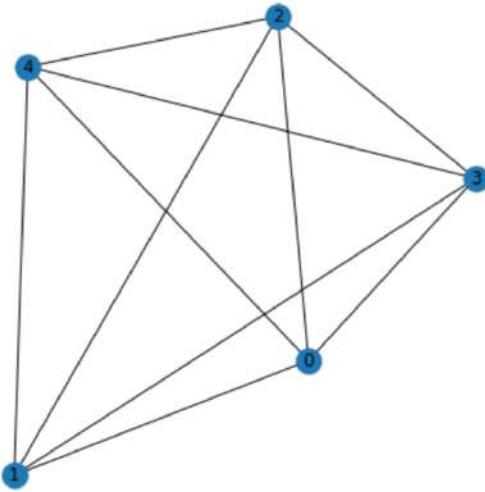
Şekil 6.6.2. Traveling Salesperson, 2. Problem

Şekil 6.6.2'de ikinci problemi temsil eden çizgeyi gözlemleyebiliriz. Bu çizge, dört adet birimden ve bu birimleri birbirine bağlayan altı adet bağlantıdan oluşmaktadır. Bağlantıların ağırlığı 10 ve 50 arasında rastgele olacak şekilde belirlenmiştir.

```
In [220]: tsp3 = nx.complete_graph(5)

for (u,v,w) in tsp3.edges(data=True):
    w['weight'] = random.randint(10,50)

plt.figure(figsize=(5,5))
nx.draw(tsp3, with_labels=True, font_weights='bold')
```



Şekil 6.6.3. Traveling Salesperson, 3. Problem

Şekil 6.6.3'de Traveling Salesperson problemi için oluşturduğumuz üçüncü ve son problemi gözlemleyebiliriz. Bu problemde beş adet birim ve bu birimleri birbirine bağlayan on adet bağlantı noktası bulunmaktadır. Önceki problemlerde olduğu gibi bu problemde de bağlantı noktalarının ağırlıkları 10 ve 50 arasında rastgele olacak şekilde belirlenmiştir.

Birinci Problem - QPU ile Çözümleme

```
In [221]: c_start = time.time()

qpu_result_a = dnx.traveling_salesperson(tsp1, qpu_sampler)

c_end = time.time()
|
qpu_result_time = (c_end - c_start)

print("1. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)
1. PROBLEM - QPU ÇÖZÜM SÜRESİ:  5.453104019165039
```

Şekil 6.6.4. Traveling Salesperson, 1. Problemin QPU ile Çözümlenmesi

Şekil 6.6.4'de birinci problemin QPU ile çözümlenmesinin ~5.45 saniye sürdüğü gözlenmektedir.

Birinci Problem - CPU ile Çözümleme

```
In [222]: c_start = time.time()

cpu_result_a = dnx.traveling_salesperson(tsp1, cpu_sampler)

c_end = time.time()

cpu_result_time = (c_end - c_start)

print("1. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)

1. PROBLEM - CPU ÇÖZÜM SÜRESİ: 0.003008127212524414
```

Şekil 6.6.5. Traveling Salesperson, 1. Problemin CPU ile Çözümlenmesi

Şekil 6.6.5’de ise birinci problemin CPU ile çözümlenmesinin ~0.003 saniye sürdüğünü gözlemliyoruz. Bu durumda, ilk problemin boyutunda, CPU’nun QPU’ya kıyasla daha hızlı olduğunu söylemek mümkündür. Devam eden bölümde, artan problem boyutunun CPU performansını nasıl etkilediğini ikinci ve üçüncü problem üzerinden gözlemliyoruz.

İkinci Problem - QPU ile Çözümleme

```
In [223]: c_start = time.time()

qpu_result_b = dnx.traveling_salesperson(tsp2, qpu_sampler)

c_end = time.time()
|
qpu_result_time = (c_end - c_start)

print("2. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)

2. PROBLEM - QPU ÇÖZÜM SÜRESİ: 6.469345808029175
```

Şekil 6.6.6. Traveling Salesperson, 2. Problemin QPU ile Çözümlenmesi

Şekil 6.6.6’da ikinci problemin QPU ile çözümlenmesinin ~6.47 saniye sürdüğünü gözlemliyoruz. İlk problem ile çözüm süreleri arasında anlamlı bir fark saptanmamıştır. Mevcut 1 saniyelik farkın, ağ kaynaklı olduğu düşünülmektedir.

İkinci Problem - CPU ile Çözümleme

```
In [224]: c_start = time.time()

cpu_result_b = dnx.traveling_salesperson(tsp2, cpu_sampler)
|
c_end = time.time()

cpu_result_time = (c_end - c_start)

print("2. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)
2. PROBLEM - CPU ÇÖZÜM SÜRESİ:  0.15728998184204102
```

Şekil 6.6.7. Traveling Salesperson, 2. Problemin CPU ile Çözümlenmesi

Şekil 6.6.7'de ise ikinci problemin CPU tarafından çözümlenme süresinin ~0.15 saniye olduğunu gözlemliyoruz. Bu süre, birinci problemin çözüm süresine göre daha yüksektir. Ancak, QPU'nun çözüm süresi ile kıyaslandığında CPU, hala daha hızlı performans göstermektedir.

Üçüncü Problem - QPU ile Çözümleme

```
In [225]: c_start = time.time()

qpu_result = dnx.traveling_salesperson(tsp3, qpu_sampler)

c_end = time.time()

qpu_result_time = (c_end - c_start)
|
print("3. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)
3. PROBLEM - QPU ÇÖZÜM SÜRESİ:  8.27685022354126
```

Şekil 6.6.8. Traveling Salesperson, 3. Problemin QPU ile Çözümü

Şekil 6.6.8 incelendiğinde, üçüncü problemin QPU tarafından çözümlenme süresinin ~8.27 saniye olduğu dikkat çekmektedir. Süre artışının sebebi problem boyutunun büyümelerinden çok, yine ağırlıkları kaynaklıdır.

Üçüncü Problem - CPU ile Çözümleme

```
In [226]: c_start = time.time()

cpu_result = dnx.traveling_salesperson(tsp3, cpu_sampler)
|
c_end = time.time()

cpu_result_time = (c_end - c_start)

print("3. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)

3. PROBLEM - CPU ÇÖZÜM SÜRESİ:  184.03486275672913
```

Şekil 6.6.9. Traveling Salesperson, 3. Problemin CPU ile Çözümü

Şekil 6.6.9'u incelediğimizde, CPU'nun üçüncü problemi çözümlemesi için geçen sürenin ~184.03 saniye olduğunu görüyoruz. Bu süre birinci ve ikinci problemin çözüm süreleri dikkate alındığında oldukça yüksektir. Bu sebeple, problemin sahip olduğu boyutta QPU'nun CPU'ya kıyasla daha hızlı olduğunu söyleyebiliriz.

Birinci Problem - Çözümlerin Karşılaştırılması

```
In [227]: print("1. PROBLEM - QPU SONUÇ: ", qpu_result_a)
print("1. PROBLEM - CPU SONUÇ: ", cpu_result_a)

1. PROBLEM - QPU SONUÇ:  [1, 1, 1]
1. PROBLEM - CPU SONUÇ:  [0, 1, 2]
```

İkinci Problem - Çözümlerin Karşılaştırılması

```
In [228]: print("2. PROBLEM - QPU SONUÇ: ", qpu_result_b)
print("2. PROBLEM - CPU SONUÇ: ", cpu_result_b)

2. PROBLEM - QPU SONUÇ:  [None, None, None, 3]
2. PROBLEM - CPU SONUÇ:  [3, 1, 0, 2]
```

Üçüncü Problem - Çözümlerin Karşılaştırılması

```
In [229]: print("3. PROBLEM - QPU SONUÇ: ", qpu_result)
print("3. PROBLEM - CPU SONUÇ: ", cpu_result)

3. PROBLEM - QPU SONUÇ:  [1, 2, 1, 2, None]
3. PROBLEM - CPU SONUÇ:  [0, 1, 2, 3, 4]
```

Şekil 6.6.10. Traveling Salesperson, QPU ve CPU Çözümlerinin Karşılaştırılması

Şekil 6.6.10 incelendiğinde QPU'dan alınan çözümlerin tutarsız olduğu görülmektedir. Bu tutarsızlıkların net sebebi çalışma süresince net olarak anlaşılamamıştır. İleri çalışmalarda çoklu denemeler gibi belirli faktörler kontrol altında tutularak tutarlılıktaki değişimler ölçümlenebilir.

6.7 Structural Imbalance Problemi

Bu bölümde Structural Imbalance/Yapısal Dengesizlik problemine yönelik oluşturduğumuz üç örnek çizgeyi D-Wave QPU ve klasik CPU kullanarak çözümledik. Çözümlemelerin gerçekleştirilme sürelerine ek olarak elde ettiğimiz sonuçları da raporlandırdık.

```
Birinci Problem
In [261]: sim1 = nx.complete_graph(5)
sim1.add_edges_from([(u, v, {'sign': 2*random.randint(0, 1) - 1}) for u, v in sim1.edges])

nx.relabel_nodes(sim1,
                 {0: 'Ayşe', 1: 'Barış', 2: 'Ceren', 3: 'Deniz', 4: 'Eda'},
                 copy=False)

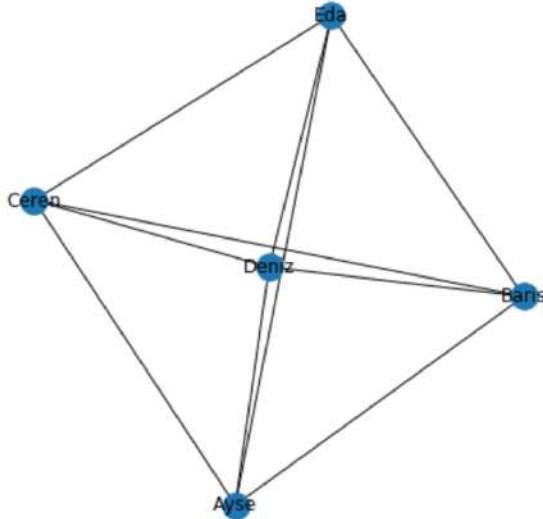
print('Positif (Arkadaşçılı) ilişkiler: \n\n'.join(
    list(x + " & " + y for (x, y, sign) in sim1.edges(data='sign') if (sign == 1))))
print('Negatif (Düşmançılı) ilişkiler: \n\n'.join(
    list(x + " & " + y for (x, y, sign) in sim1.edges(data='sign') if (sign == -1)))

Positif (Arkadaşçılı) ilişkiler:
Ayşe & Ceren
Barış & Ceren
Ceren & Deniz
Negatif (Düşmançılı) ilişkiler:
Ayşe & Barış
Ayşe & Deniz
Ayşe & Eda
Barış & Deniz
Barış & Eda
Ceren & Eda
Deniz & Eda
```

Şekil 6.7.1. Structural Imbalance, 1. Problem

Şekil 6.7.1'de, Structural Imbalance problemini örneklendirmek için oluşturduğumuz üç adet çizgeden ilkinin tanımını gözlemliyoruz. Bu tanımda, öncelikli olarak beş birimden ve bu beş birimin arasındaki toplam on adet bağlantıdan oluşan çizgemi tanımladık. Ardından bu bağlantıların değerlerini, yani çizgemi侦deki birimlerin birbirleri ile olan ilişkilerinin pozitif/negatiflik durumunu rastgele olarak belirledik. Çizgemi侦de bulunan birimleri kişiler olarak örnekleme adına onlara isim verdik. Sonrasında belirlenen rastgele ilişkilerin ekranda görüntülenmesini sağladık.

```
In [262]: plt.figure(figsize=(5,5))
nx.draw(sim1, with_labels=True, font_weights='bold')
```



Şekil 6.7.2. Structural Imbalance, 1. Problem Çizge

İkinci Problem

```
In [263]: sim2 = nx.complete_graph(15)
sim2.add_edges_from([(u, v, {'sign': 2*random.randint(0, 1) - 1}) for u, v in sim2.edges])
nx.relabel_nodes(sim2,
                 {0: 'Ayse', 1: 'Baris', 2: 'Ceren', 3: 'Deniz', 4: 'Eda', 5: 'Fahrettin', 6: 'Gaye',
                  7: 'Hulya', 8: 'Idil', 9: 'Jale', 10: 'Kadir', 11: 'Lale', 12: 'Melis', 13: 'Nedim', 14: 'Onur'},
                 copy=False)

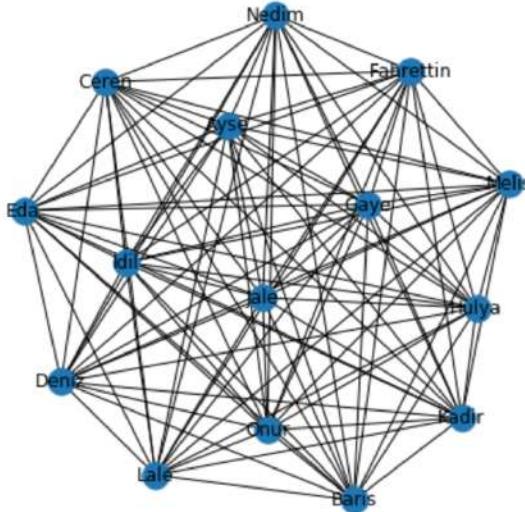
print('Pozitif (Arkadaşçılı) ilişkiler: \n\n'.join(
    list(x + " & " + y for (x, y, sign) in sim2.edges(data='sign') if (sign == 1))))
print('Negatif (Düşmancılı) ilişkiler: \n\n'.join(
    list(x + " & " + y for (x, y, sign) in sim2.edges(data='sign') if (sign == -1))))
```

```
Pozitif (Arkadaşçılı) ilişkiler:
Ayse & Baris
Ayse & Deniz
Ayse & Eda
Ayse & Gaye
Ayse & Hulya
Ayse & Melis
Ayse & Nedim
Baris & Ceren
Baris & Deniz
Baris & Fahrettin
Baris & Gaye
Baris & Hulya
Baris & Melis
Baris & Onur
Ceren & Deniz
Ceren & Fahrettin
Ceren & Hulya
Ceren & Idil
Ceren & Lale
Ceren & Melis
Ceren & Nedim
Ceren & Onur
Deniz & Fahrettin
Deniz & Gaye
Deniz & Hulya
Deniz & Melis
Deniz & Nedim
Deniz & Onur
Eda & Fahrettin
Eda & Gaye
Eda & Hulya
Eda & Melis
Eda & Nedim
Eda & Onur
Fahrettin & Gaye
Fahrettin & Hulya
Fahrettin & Melis
Fahrettin & Nedim
Fahrettin & Onur
Gaye & Hulya
Gaye & Melis
Gaye & Nedim
Gaye & Onur
Hulya & Melis
Hulya & Nedim
Hulya & Onur
Idil & Lale
Idil & Melis
Idil & Nedim
Idil & Onur
Jale & Lale
Jale & Melis
Jale & Nedim
Jale & Onur
Kadir & Lale
Kadir & Melis
Kadir & Nedim
Kadir & Onur
Lale & Melis
Lale & Nedim
Lale & Onur
Melis & Nedim
Melis & Onur
Nedim & Onur
```

Şekil 6.7.3. Structural Imbalance, 2. Problem

Şekil 6.7.3'te birinci problemdekine benzer şekilde, çizgede bulunan toplamda on beş adet birimin birbiri ile olan pozitif/negatif ilişki durumu rastgele olacak şekilde belirlenmiş; çizgede var olan birimlere isimler verilmiş; ardından rastgele olarak belirlenen bu ilişkiler ekranda görüntülenmiştir.

```
In [264]: plt.figure(figsize=(5,5))
nx.draw(sim2, with_labels=True, font_weights='bold')
```



Şekil 6.7.4. Structural Imbalance, 2. Problem Çizge

Üçüncü Problem

```
In [265]: sim3 = nx.complete_graph(25)
sim3.add_edges_from([(u, v, {'sign': 2*random.randint(0, 1) - 1}) for u, v in sim3.edges])

nx.relabel_nodes(sim3,
                 {0: 'Ayse', 1: 'Baris', 2: 'Ceren', 3: 'Deniz', 4: 'Eda', 5: 'Fahrettin', 6: 'Gaye',
                  7: 'Hulya', 8: 'Idil', 9: 'Jale', 10: 'Kadir', 11: 'Lale', 12: 'Melis', 13: 'Nedim', 14: 'Onur',
                  15: 'Pelin', 16: 'Rasin', 17: 'Selim', 18: 'Tolga', 19: 'Ulvi', 20: 'Volkan', 21: 'Yavuz',
                  22: 'Zelihha', 23: 'Anil', 24: 'Beril'},
                 copy=False)

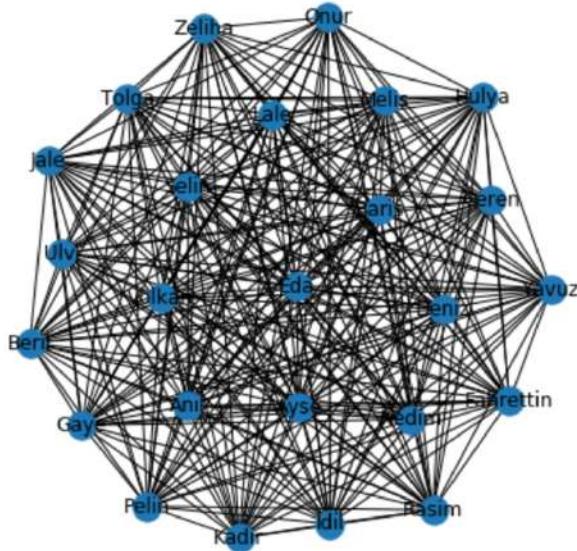
print('Positif (Arkadaşcıl) ilişkiler: \n' + '\n'.join(
    list(x + " & " + y for (x, y, sign) in sim3.edges(data='sign') if (sign == 1))))
print('Negatif (Düşmancıl) ilişkiler: \n' + '\n'.join(
    list(x + " & " + y for (x, y, sign) in sim3.edges(data='sign') if (sign == -1))))
```

Selin & Ulvi
Selin & Zelihha
Tolga & Ulvi
Tolga & Beril
Ulvi & Volkan
Ulvi & Anil
Ulvi & Beril
Volkan & Yavuz
Volkan & Beril
Yavuz & Zelihha
Zelihha & Anil
Negatif (Düşmancıl) ilişkiler:
Ayse & Baris
Ayse & Ceren
Ayse & Deniz
Ayse & Gaye
Ayse & Melis
Ayse & Onur
Ayse & Rasin
Ayse & Selim

Şekil 6.7.5. Structural Imbalance, 3. Problem

Şekil 6.7.5'de Structural Imbalance problemi için örnek olarak oluşturduğumuz üçüncü ve son çizgenin tanımını gözlemlemekteyiz. Burada da birinci ve ikinci probleme benzer şekilde isim tanımlaması gerçekleştirilmiş, birimler arası bağlantılar rastgele değerler atanmış ve ekranda görüntülenmiştir. Farkı ise bu problemin yirmi beş adet birimden/kişiden meydana gelmesidir.

```
In [266]: plt.figure(figsize=(5,5))
nx.draw(sim3, with_labels=True, font_weights='bold')
```



Şekil 6.7.6. Structural Imbalance, 3. Problem Çizge

Birinci Problem - QPU ile Çözümleme

```
In [274]: c_start = time.time()
qpu_imbalance_a, qpu_bicoloring_a = dnx.structural_imbalance(sim1, qpu_sampler)
c_end = time.time()
qpu_result_time = (c_end - c_start)
print("1. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)

1. PROBLEM - QPU ÇÖZÜM SÜRESİ:  5.223376989364624
```

Şekil 6.7.7. Structural Imbalance, 1. Problemin QPU ile Çözümlenmesi

Şekil 6.7.7'de görülebileceği üzere, birinci problemin QPU tarafından çözümlenmesi ~5.22 saniyede gerçekleştirılmıştır.

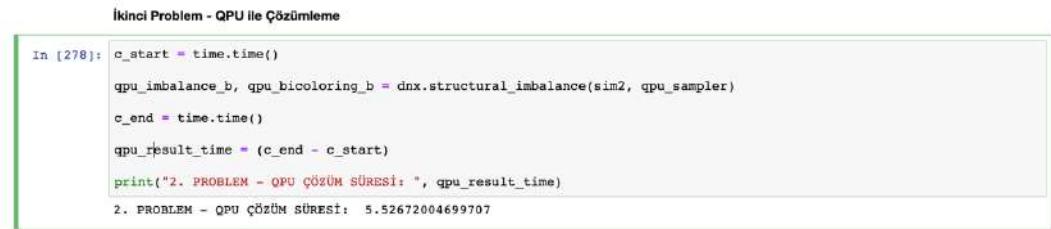
Birinci Problem - CPU ile Çözümleme

```
In [275]: c_start = time.time()
cpu_imbalance_a, cpu_bicoloring_a = dnx.structural_imbalance(sim1, cpu_sampler)
c_end = time.time()
cpu_result_time = (c_end - c_start)
print("1. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)

1. PROBLEM - CPU ÇÖZÜM SÜRESİ:  0.0013730525970458984
```

Şekil 6.7.8. Structural Imbalance, 1. Problemin CPU ile Çözümlenmesi

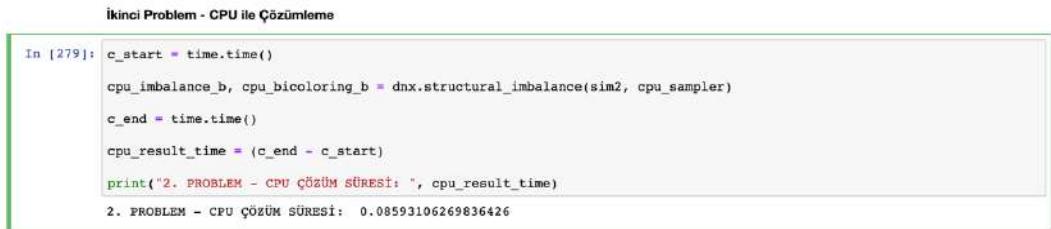
Şekil 6.7.8'de gözlemlenildiği üzere, CPU'nun birinci problemi çözümlemesi ise ~0.0013 saniyede gerçekleşmiştir. Bu durumda birinci problemin sahip olduğu problem boyutunda CPU, QPU'ya kıyasla daha hızlı faaliyet gösterebilmektedir. İkinci ve üçüncü problemlerde daha büyük boyutlarla çalışıp CPU performansında oluşan değişimleri gözlemledik.



```
İkinci Problem - QPU ile Çözümleme
In [278]: c_start = time.time()
qpu_imbalance_b, qpu_bicoloring_b = dnx.structural_imbalance(sim2, qpu_sampler)
c_end = time.time()
qpu_result_time = (c_end - c_start)
print("2. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)
2. PROBLEM - QPU ÇÖZÜM SÜRESİ:  5.52672004699707
```

Şekil 6.7.9. Structural Imbalance, 2. Problemin QPU ile Çözümlenmesi

Şekil 6.7.9 incelendiğinde, ikinci problemin QPU tarafından ~5.53 saniyede çözümlendiği sonucuna ulaşılabilir. Birinci problemin çözüm performansı ile kıyaslandığında arada neredeyse hiçbir anlamlı fark bulunmamaktadır.



```
İkinci Problem - CPU ile Çözümleme
In [279]: c_start = time.time()
cpu_imbalance_b, cpu_bicoloring_b = dnx.structural_imbalance(sim2, cpu_sampler)
c_end = time.time()
cpu_result_time = (c_end - c_start)
print("2. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)
2. PROBLEM - CPU ÇÖZÜM SÜRESİ:  0.08593106269836426
```

Şekil 6.7.10 Structural Imbalance, 2. Problemin CPU ile Çözümlenmesi

Şekil 6.7.10'da ise ikinci problemin CPU tarafından çözümlenmesinin ~0.086 saniyede gerçekleştiğini gözlemliyoruz. Her ne kadar ilk problemin çözüm süresine kıyasla bir artış olduğu söylenebilirse de, QPU ile kıyaslandığında CPU hala daha hızlı performans göstermektedir.

Üçüncü Problem - QPU ile Çözümleme

```
In [282]: c_start = time.time()

qpu_imbalance, qpu_bicoloring = dnx.structural_imbalance(sim3, qpu_sampler)

c_end = time.time()

qpu_result_time = (c_end - c_start)

print("3. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)

3. PROBLEM - QPU ÇÖZÜM SÜRESİ:  9.294797897338867
```

Şekil 6.7.11. Structural Imbalance, 3. Problemin QPU ile Çözümlenmesi

Şekil 6.7.11'de üçüncü problemin QPU ile çözümlenmesinin ~9.29 saniyede gerçekleştiğini gözlemliyoruz. Diğer problemlere kıyasla birkaç saniye daha uzun sürede işlemin tamamlanmasının sebebinin ağır gecikmesi olduğunu düşünmekteyiz.

Üçüncü Problem - CPU ile Çözümleme

```
In [283]: c_start = time.time()

cpu_imbalance, cpu_bicoloring = dnx.structural_imbalance(sim3, cpu_sampler)

c_end = time.time()

cpu_result_time = (c_end - c_start)

print("3. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)

3. PROBLEM - CPU ÇÖZÜM SÜRESİ:  307.7670600414276
```

Şekil 6.7.12. Structural Imbalance, 3. Problemin CPU ile Çözümlenmesi

Şekil 6.7.12'de üçüncü problemin CPU ile çözümlenmesinin ~307.76 saniye olduğunu inceleyebiliriz. Birinci ve ikinci problemle kıyaslandığında oldukça ciddi bir süre artışı olmak ile birlikte, QPU performansı ile kıyaslandığı durumda QPU'nun bu problem boyutunda CPU'ya kıyasla daha avantajlı ve hızlı olduğu söylenebilmektedir.

Birinci Problem - Çözümlerin Karşılaştırılması

```
In [276]: for edge in sim1.edges:
    sim1.edges[edge]['frustrated'] = edge in qpu_imbalance_a
for node in sim1.nodes:
    sim1.nodes[node]['color'] = qpu_bicoloring_a[node]

print("1.PROBLEM (QPU):")
print('GRUP A: \n\t' + '\n\t'.join(list(person for (person, color) in qpu_bicoloring_a.items() if (color == 0))))
print('GRUP B: \n\t' + '\n\t'.join(list(person for (person, color) in qpu_bicoloring_a.items() if (color == 1))))
print('DENGESİZ/INSTABİL İLİŞKİLER: \n\t' + '\n\t'.join(list(x + " & " + y for (x, y) in qpu_imbalance_a.keys())))

1.PROBLEM (QPU):
GRUP A:
    Eda
GRUP B:
    Ayse
    Baris
    Ceren
    Deniz
DENGESİZ/INSTABİL İLİŞKİLER:
    Ayse & Baris
    Ayse & Deniz
    Baris & Deniz
```

Şekil 6.7.13. Structural Imbalance, 1. Problem, QPU Sonuç

Şekil 6.7.13'de birinci problemin QPU tarafından elde edilmiş çözümü görüntülenmektedir. Grup A ve Grup B olarak ayrılan birimlere ek olarak, dengesiz ilişkiler de ekranda görüntülenmektedir.

```
In [277]: for edge in sim1.edges:
    sim1.edges[edge]['frustrated'] = edge in cpu_imbalance_a
for node in sim1.nodes:
    sim1.nodes[node]['color'] = cpu_bicoloring_a[node]

print("1.PROBLEM (CPU):")
print("GRUP A: \n\t".join(list(person for (person, color) in cpu_bicoloring_a.items() if (color == 0))))
print("GRUP B: \n\t".join(list(person for (person, color) in cpu_bicoloring_a.items() if (color == 1))))
print('DENGESIZ/INSTABİL İLİŞKİLER: \n\t'.join(list(x + " & " + y for (x, y) in cpu_imbalance_a.keys())))

1.PROBLEM (CPU):
GRUP A:
Ayse
Baris
Ceren
Deniz
GRUP B:
Eda
DENGESIZ/INSTABİL İLİŞKİLER:
Ayse & Baris
Ayse & Deniz
Baris & Deniz
```

Şekil 6.7.14. Structural Imbalance, 1. Problem CPU Sonuç

Şekil 6.7.14'de ise birinci problemin CPU tarafından elde edilmiş çözümü görüntülenmektedir. QPU tarafından elde edilmiş çözümle bire bir aynıdır ve tutarlıdır. Daha büyük problem boyutlarında QPU ve CPU çözümlerinin tutarlı olup olmadığı ise ilerleyen kısımlarda gösterilecektir.

```
In [280]: for edge in sim2.edges:
    sim2.edges[edge]['frustrated'] = edge in qpu_imbalance_b
for node in sim2.nodes:
    sim2.nodes[node]['color'] = qpu_bicoloring_b[node]

print("2.PROBLEM (QPU):")
print("GRUP A: \n\t".join(list(person for (person, color) in qpu_bicoloring_b.items() if (color == 0))))
print("GRUP B: \n\t".join(list(person for (person, color) in qpu_bicoloring_b.items() if (color == 1))))
print('DENGESIZ/INSTABİL İLİŞKİLER: \n\t'.join(list(x + " & " + y for (x, y) in qpu_imbalance_b.keys())))

2.PROBLEM (QPU):
GRUP A:
Ayse
Baris
Deniz
Fahrettin
Gaye
Hulya
Onur
GRUP B:
Ceren
Eda
Jale
Kadir
Lale
Melis
Nedim
fdil
DENGESIZ/INSTABİL İLİŞKİLER:
Ayse & Eda
Ayse & Fahrettin
Ayse & Melis
Ayse & Nedim
Ayse & Onur
Baris & Ceren
Baris & Melis
Ceren & Deniz
```

Şekil 6.7.15. Structural Imbalance, 2. Problem QPU Sonuç

Şekil 6.7.15'de birinci probleme kıyasla daha karmaşık bir çizge olan ikinci problemin çözümü ekranда görüntülenmektedir. Karmaşık ilişki sayısı daha fazladır. Tüm ilişkiler şekil üzerinde görüntülenmemiştir ancak Ek1 kısmında detaylıca belirtilmektedir.

```
In [281]: for edge in sim2.edges:
    sim2.edges[edge]['frustrated'] = edge in cpu_imbalance_b
for node in sim2.nodes:
    sim2.nodes[node]['color'] = cpu_bicoloring_b[node]

print("2.PROBLEM (CPU):")
print('GRUP A: \n\t' + '\n\t'.join(list(person for (person, color) in cpu_bicoloring_b.items() if (color == 0))))
print('GRUP B: \n\t' + '\n\t'.join(list(person for (person, color) in cpu_bicoloring_b.items() if (color == 1))))
print('DENGESİZ/INSTABİL İLİŞKİLER: \n\t' + '\n\t'.join(list(x + " & " + y for (x, y) in cpu_imbalance_b.keys())))

2.PROBLEM (CPU):
GRUP A:
Ayse
Baris
Eda
Fahrettin
Gaye
Hulya
Melis
Idil
GRUP B:
Ceren
Deniz
Jale
Kadir
Lale
Nedim
Onur
DENGESİZ/INSTABİL İLİŞKİLER:
Ayse & Deniz
Ayse & Fahrettin
Ayse & Idil
Ayse & Nedim
Baris & Ceren
Baris & Deniz
Baris & Eda
Baris & Idil
Baris & Onur
Ceren & Fahrettin
Ceren & Hulya
Ceren & Idil
Ceren & Jale
Deniz & Hulya
```

Şekil 6.7.16. Structural Imbalance, 2. Problem CPU Sonuç

Şekil 6.7.16'da, ikinci probleme CPU'nun getirdiği çözümü gözlemlemekteyiz. QPU ile kıyaslandığında daha farklı bir ayrıştırma uyguladığını söylemek mümkündür. Frustrated/kafası karışık ilişkiler arasında da doğal olarak farklılıklar bulunmaktadır. Hangi ayrıştırmanın daha verimli olduğunu söylemek için daha ileri araştırmalar yapılması önerilir. Tüm ilişkiler şekil üzerinde görüntülenmemiştir ancak Ek1 kısmında detaylıca belirtilmektedir.

```

Üçüncü Problem - Çözümlerin Karşılaştırılması

In [284]: for edge in sim3.edges:
    sim3.edges[edge]['frustrated'] = edge in gpu_imbalance
for node in sim3.nodes:
    sim3.nodes[node]['color'] = gpu_bicoloring[node]

print("3.PROBLEM (QPU):")
print('GRUP A: \n\t' + '\n\t'.join(list(person for (person, color) in gpu_bicoloring.items() if (color == 0))))
print('GRUP B: \n\t' + '\n\t'.join(list(person for (person, color) in gpu_bicoloring.items() if (color == 1))))
print('DENGESİZ/INSTABİL İLİŞKİLER: \n\t' + '\n\t'.join(list(x + " & " + y for (x, y) in gpu_imbalance.keys())))

3.PROBLEM (QPU):
GRUP A:
    Anıl
    Ayşe
    Beril
    Eda
    Hulya
    Jale
    Lale
    Melis
    Nedim
    Onur
    Volkan
    Yavuz
    idil

GRUP B:
    Barış
    Ceren
    Deniz
    ...

```

Şekil 6.7.17. Strucural Imbalance, 3. Problem QPU Sonuç

Şekil 6.7.17'de üçüncü ve son probleme ait QPU tarafından getirilen çözümü inceleyebiliriz. Tüm ilişkiler şekil üzerinde görüntülenmemiştir ancak Ek1 kısmında detaylıca belirtilmektedir.

```

In [285]: for edge in sim3.edges:
    sim3.edges[edge]['frustrated'] = edge in cpu_imbalance
for node in sim3.nodes:
    sim3.nodes[node]['color'] = cpu_bicoloring[node]

print("3.PROBLEM (CPU):")
print('GRUP A: \n\t' + '\n\t'.join(list(person for (person, color) in cpu_bicoloring.items() if (color == 0))))
print('GRUP B: \n\t' + '\n\t'.join(list(person for (person, color) in cpu_bicoloring.items() if (color == 1))))
print('DENGESİZ/INSTABİL İLİŞKİLER: \n\t' + '\n\t'.join(list(x + " & " + y for (x, y) in cpu_imbalance.keys())))

Jale & Tolga
Jale & Ulvi
Jale & Volkan
Jale & Yavuz
Jale & Anıl
Kadir & Nedim
Kadir & Onur
Kadir & Pelin
Kadir & Rasim
Kadir & Volkan
Kadir & Yavuz
Kadir & Anıl
Lale & Onur
Lale & Ulvi
Lale & Anıl
Melis & Nedim
Melis & Onur
Melis & Pelin
Melis & Beril
Nedim & Rasim

```

Şekil 6.7.18: Structural Imbalance, 3. Problem CPU Sonuç

Şekil 6.7.18'de üçüncü probleme CPU'nun bulduğu çözümü gözlemleyebiliriz. Bu problemde de ikinci probleme benzer şekilde ayırtılmada farklılıklar söz konusudur. Bu farklı ayırtmalar, benzer verimlilik düzeyinde olabileceği gibi biri diğerine kıyasla daha avantajlı da olabilir. Bunu anlamanın en etkin yolu, QPU ve CPU arasındaki tutarlılık farklarını test etme amaçlı bir ileri çalışma gerçekleştirmektir. Tüm ilişkiler şekil üzerinde görüntülenmemiştir ancak Ek1 kısmında detaylıca belirtilmektedir.

7. D-WAVE QPU/CPU HİBRİTLEME VE CPU İLE ÖRNEK ÇÖZÜMLER

Kuantum Bilgisayarlar günümüzde hala gelişme sürecinde olan, klasik bilgisayarlar ile kıyaslandığında oldukça yeni sayılabilecek cihazlardır. Bu sebeple, mevcut kuantum bilgisayarların tam anlamıyla hatasız ve yüksek tutarlılıkla çalıştığını iddia etmek çok doğru olmaz. Bu gelişme süreci boyunca, QPU'ların klasik CPU'lar ile desteklenmesi sağlanarak, işlem sürelerinde olumlu geri dönüşler alınabileceği düşüncesi ile D-Wave tarafından QPU/CPU Hibrit çözümleyiciler hizmete sunulmuştur. Bu çözümleyiciler QPU ve CPU'ların avantajlarını belirli noktalarda birleştirerek, daha büyük boyutlu belirli problemlerin çözülebilmesine olanak sağlayabilmektedir.

Çalışmamızın bu bölümünde, D-Wave QPU/CPU hibrit çözümleyicileri kullanarak çeşitli yönyelem problemlerine çözümler getireceğiz. Bu çözümlemeler aynı zamanda QPU ve CPU kullanılarak da gerçekleştirilecek. Elde edilen çözümlemelerin süreleri, elde edilen sonuçlar ile birlikte raporlanıp görüntülenecek. Bu bölümde, her yönyelem problemi için örnek bir çizge tanımlanacak.

```
In [287]: from dwave.system import LeapHybridSampler
hybrid_sampler = LeapHybridSampler()
```

Şekil 7.1. D-Wave QPU/CPU Hibrit Çözümleyicinin Import Edilmesi

Şekil 7.1'de görüleceği gibi, D-Wave kütüphanesine ait LeapHybridSampler çözümleyicisini, tipki QPU ve CPU ünitelerine ait çözümleyicileri yarattığımız gibi yaratıyoruz.

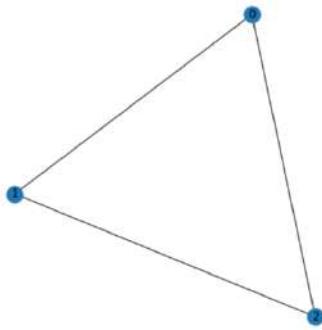
7.1 Minimum Vertex Cover Problemi

Bu bölümde Minimum Vertex Cover probleminin QPU/CPU ile çözümlenmesi ve performans ölçütlerinin gerçekleştirilebilmesi için Network-X kütüphanesi kullanarak örnek bir problem yarattık.

```
In [301]: hvcl = nx.complete_graph(3)
for (u,v,w) in hvcl.edges(data=True):
    w['weight'] = random.randint(0,20)

plt.figure(figsize=(5,5))
nx.draw(hvcl, with_labels=True, font_weights='bold')

/Users/egedursun/opt/anaconda3/lib/python3.7/site-packages/networkx/drawing/nx_pylab.py:579: MatplotlibDeprecationWarning:
The iterable function was deprecated in Matplotlib 3.1 and will be removed in 3.3. Use np.iterable instead.
if not cb.iterable(width):
```



Şekil 7.1.1. Minimum Vertex Cover, Problem

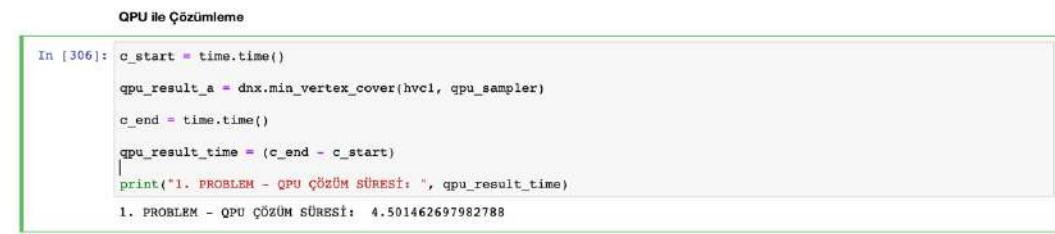
Şekil 7.1.1'de incelenileceği gibi, Minimum Vertex Cover çözümlemesi için oluşturulan çizge üç adet birimden ve bu birimleri birbirine bağlayan üç adet bağlantıdan meydana gelmektedir. Bağlantıların ağırlıkları 0 ve 20 arasında rastgele sayılar olarak belirlenmiştir.

QPU/CPU Hibrit ile Çözümleme

```
In [304]: c_start = time.time()
hybrid_result_a = dnx.min_vertex_cover(hvcl, hybrid_sampler)
c_end = time.time()
hybrid_result_time = (c_end - c_start)
print("1. PROBLEM - QPU/CPU HİBRİT ÇÖZÜM SÜRESİ: ", hybrid_result_time)
1. PROBLEM - QPU/CPU HİBRİT ÇÖZÜM SÜRESİ:  3676.800828933716
```

Şekil 7.1.2. Minimum Vertex Cover, Problemin QPU/CPU Hibrit ile Çözümlenmesi

Şekil 7.1.2'de görülebileceği üzere Minimum Vertex Cover problemini örneklemek için tasarladığımız çizgenin QPU/CPU hibriti tarafından çözümlenmesi ~3676 saniye sürmüştür. Sürenin neden bu kadar yüksek olduğu, problemin yapısından mı; hibrit çözümlemeye herhangi bir sebep ile uyumsuzluktan mı; yoksa ağ gecikmelerinden mi kaynaklandığı çözümlenmemiştir. Problemin boyutu düşünüldüğünde kabul edilebilir bir çözümleme süresi değildir.



```

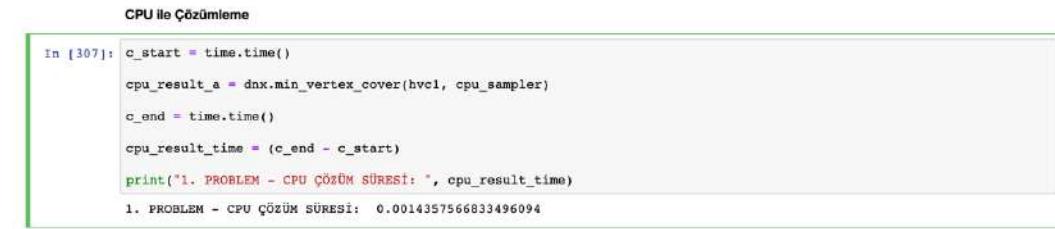
QPU ile Çözümleme

In [306]: c_start = time.time()
          gpu_result_a = dnx.mia_vertex_cover(hvcl, qpu_sampler)
          c_end = time.time()
          gpu_result_time = (c_end - c_start)
          print("1. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", gpu_result_time)
1. PROBLEM - QPU ÇÖZÜM SÜRESİ:  4.501462697982788

```

Şekil 7.1.3. Minimum Vertex Cover, Problemin QPU ile Çözümlenmesi

Şekil 7.1.3 incelendiğinde, problemin çözümlenmesinin ~4.5 saniyede gerçekleştiğini görebiliriz. QPU çözümlemesi için beklenen ortalama süreyle yakın bir değer olmak ile birlikte mevcut deneyde QPU/CPU hibritine kıyasla çok daha hızlı olduğu sonucuna ulaşılmıştır.



```

CPU ile Çözümleme

In [307]: c_start = time.time()
          cpu_result_a = dnx.min_vertex_cover(hvcl, cpu_sampler)
          c_end = time.time()
          cpu_result_time = (c_end - c_start)
          print("1. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)
1. PROBLEM - CPU ÇÖZÜM SÜRESİ:  0.0014357566833496094

```

Şekil 7.1.4. Minimum Vertex Cover, Problemin CPU ile Çözümlenmesi

Son olarak Şekil 7.1.4'de, problemin CPU tarafından ~0.0014 saniyede çözümlendiğini görüyoruz. Bu durumda mevcut problem boyutu için CPU'nun, QPU ve QPU/CPU hibritinden daha yüksek performans gösterdiğini söylemek doğru olacaktır. Daha büyük problem boyutlarında performansın nasıl değişeceğini ile ilgili ileri çalışmalar gerçekleştirilebilir.

Çözümlerin Karşılaştırılması

```
In [312]: print("QPU/CPU HİBRİT SONUÇ: ", hybrid_result_a)
print("QPU SONUÇ: ", gpu_result_a)
print("CPU SONUÇ: ", cpu_result_a)

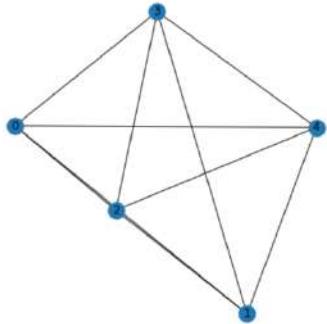
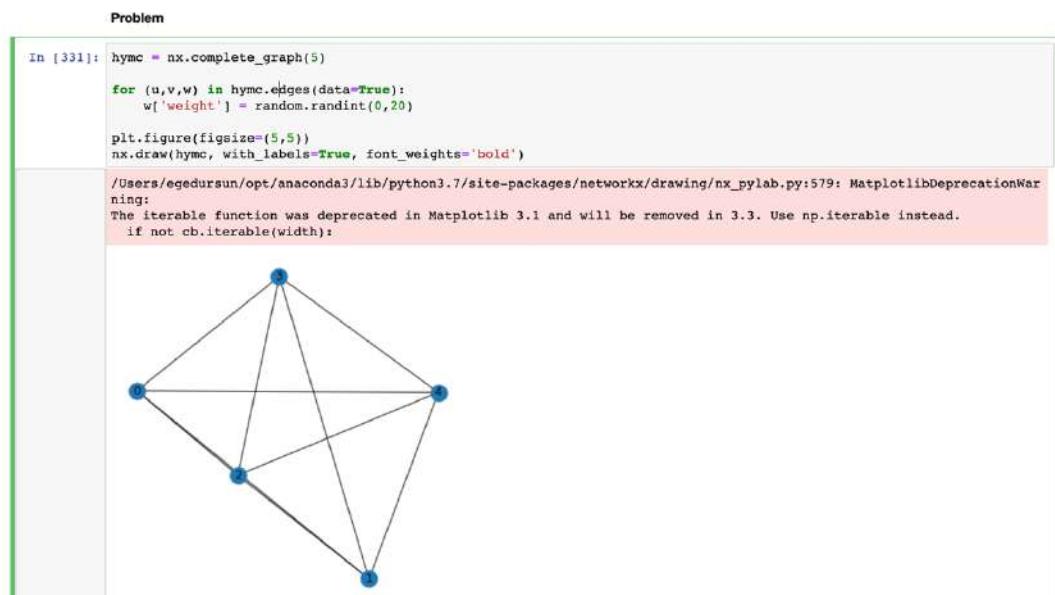
QPU/CPU HİBRİT SONUÇ: [0, 1]
QPU SONUÇ: [1, 2]
CPU SONUÇ: [1, 2]
```

Şekil 7.1.5. Minimum Vertex Cover, QPU/CPU Hibrit, QPU ve CPU Çözümlerinin Karşılaştırılması

Şekil 7.1.5’de QPU ve CPU’nun problemin çözümü konusunda ulaştıkları cevabın aynı olduğunu gözlemlerken QPU/CPU hibritinin farklı bir cevaba ulaştığını görüyoruz. Bunun sebebi, QPU/CPU hibritinin mevcut problem üzerinde iyi performans gösterememesi olabileceği gibi; önceki bölümde bahsedilen tek ölçümleme kaynaklı sebepler veya problemin çok cevaplılığı gibi durumlar da olabilir. Nedenlerin daha iyi analiz edilmesi için ileri araştırma gerekmektedir.

7.2 Map Coloring Problemi

Bu bölümde Map Coloring problemini örneklendirme amacıyla D-Wave Network-X kütüphanesi kullanarak örnek bir çizge oluşturacağız. Ardından D-Wave QPU/CPU hibrit, D-Wave QPU ve klasik CPU çözümleyiciler kullanarak problemlerin çözümlerine ulaşacağız. Son olarak bu çözümleri ve performans göstergelerini raporlandıracak ve görüntüleyeceğiz.



Şekil 7.2.1. Map Coloring, Problem

Şekil 7.2.1'de Map Coloring problemini öneklemek amacıyla oluşturduğumuz çizgeyi inceleyebilirsiniz. Bu çizge toplamda beş adet birim ve bu birimleri birbirine bağlayan on adet bağlantıdan meydana gelmektedir. Bağlantıların ağırlıkları 0 ve 20 arasında rastgele değerler ile belirlenmiştir.

```

QPU/CPU Hibrit ile Çözümleme

In [332]: c_start = time.time()
           hybrid_result_a = dnx.min_vertex_coloring(hymc, hybrid_sampler)
           c_end = time.time()
           hybrid_result_time = (c_end - c_start)
           print("QPU/CPU HIBRİT ÇÖZÜM SÜRESİ: ", hybrid_result_time)
QPU/CPU HIBRİT ÇÖZÜM SÜRESİ:  14.24190092086792

QPU/CPU Hibrit ile Çözümleme

In [332]: c_start = time.time()
           hybrid_result_a = dnx.min_vertex_coloring(hymc, hybrid_sampler)
           c_end = time.time()
           hybrid_result_time = (c_end - c_start)
           print("QPU/CPU HIBRİT ÇÖZÜM SÜRESİ: ", hybrid_result_time)
QPU/CPU HIBRİT ÇÖZÜM SÜRESİ:  14.24190092086792

```

Şekil 7.2.2. Map Coloring, Problemin QPU/CPU Hibrit ile Çözümlenmesi

Şekil 7.2.2'den anlaşılabileceği üzere, Map Coloring probleminin QPU/CPU tarafından çözümlenmesi ~ 14.24 saniye sürmüştür. Önceki problem türü ile kıyaslandığında çok kısa bir süre olmak ile birlikte kullanılabilirliği, QPU ve CPU çözümleyicilerin performanslarına bağlıdır.

```

QPU ile Çözümleme

In [333]: c_start = time.time()
           qpu_result_a = dnx.min_vertex_coloring(hymc, qpu_sampler)
           c_end = time.time()
           qpu_result_time = (c_end - c_start)
           print("QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)
QPU ÇÖZÜM SÜRESİ:  5.506086111068726

```

Şekil 7.2.3. Map Coloring, Problemin QPU ile Çözümlenmesi

Şekil 7.2.3'de Map Coloring probleminin QPU ile çözümlenmesinin ~ 5.51 saniye süregünü görüyoruz. Bu durumda QPU çözümleyicinin QPU/CPU hibrit çözümleyiciden daha hızlı performans gösterdiğini söylemek yanlış olmayacağı.

CPU ile Çözümleme

```
In [334]: c_start = time.time()
cpu_result_a = dnx.min_vertex_coloring(hymc, cpu_sampler)
c_end = time.time()
cpu_result_time = (c_end - c_start)
print("CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)
CPU ÇÖZÜM SÜRESİ: 123.29105687141418
```

Şekil 7.2.4. Map Coloring, Problemin CPU ile Çözümlenmesi

Şekil 7.2.4'de ise Map Coloring probleminin CPU tarafından çözümlenmesinin ~123.29 saniye sürdüğünü görmekteyiz. Bu durumda CPU hem QPU/CPU hibritinden, hem de QPU'dan daha yavaş performans göstermiştir. O halde bu problem özelinde en verimli olan çözümleyicinin QPU, ardından QPU/CPU hibriti olduğunu söylemek doğru olacaktır.

Çözümlerin Karşılaştırılması

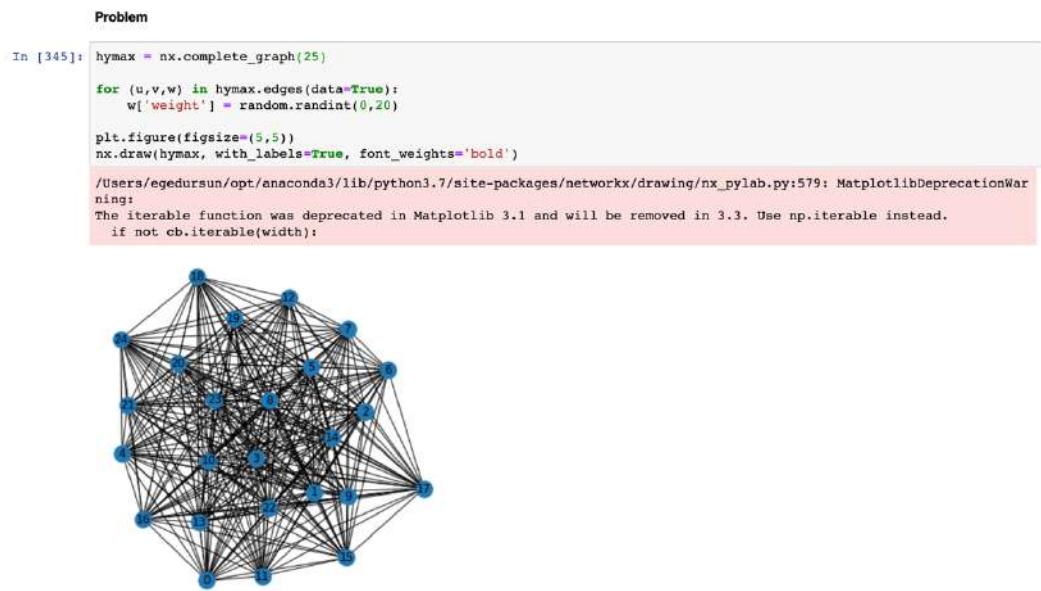
```
In [335]: print("QPU/CPU HİBRİT SONUÇ: ", hybrid_result_a)
print("QPU SONUÇ: ", qpu_result_a)
print("CPU SONUÇ: ", cpu_result_a)
QPU/CPU HİBRİT SONUÇ: {0: 4, 1: 0, 2: 1, 3: 3, 4: 2}
QPU SONUÇ: {0: 2, 1: 0, 2: 4, 4: 3}
CPU SONUÇ: {0: 3, 1: 4, 2: 1, 3: 2, 4: 0}
```

Şekil 7.2.5. Map Coloring, QPU/CPU Hibrit, QPU ve CPU Çözümlerinin Karşılaştırılması

Şekil 7.2.5'de Map Coloring probleminin QPU/CPU Hibrit, QPU ve CPU ile elde edilen çözümlemelerinin birbiri ile karşılaştırıldığını gözlemliyoruz. Diğer problemlerden alışık olduğumuz gibi burada da çözümler arasında belirli farklılıklar ve örtüşmezlikler mevcut. Bu örtüşmezliklerin sebebinin anlaşılması, QPU'ların gelecekte kullanılabileceği ticari ve akademik alanlardaki fizibilitesi açısından önem taşımaktadır.

7.3 Maximum Clique Problemi

Bu bölümde Maximum Clique probleminin örneklendirilmesi amacıyla Network-X kütüphanesi kullanılarak bir çizge oluşturduk ve bu çizgeyi D-Wave QPU/CPU Hibrit, D-Wave QPU ve klasik CPU kullanarak çözümledik. Çözümlemelerimiz sonucunda elde ettiğimiz verileri ve performans metriklerini kaydettik ve raporlandırdık.



Şekil 7.3.1. Maximum Clique, Problem

Şekil 7.3.1'de Maximum Clique problemini örneklemek için oluşturduğumuz çizgeyi gözlemleyebiliriz. Bu çizge yirmi beş adet birimden ve bu birimleri birbirine bağlayan üç yüz adet bağlantıdan oluşmaktadır. Bu bağlantıların ağırlıkları 0 ve 20 arasında rastgele olarak belirlenmektedir.

QPU/CPU Hibrit ile Çözümleme

```
In [346]: c_start = time.time()
hybrid_result_a = dnx.maximum_clique(hymax, hybrid_sampler)
c_end = time.time()
hybrid_result_time = (c_end - c_start)
print("QPU/CPU HIBRİT ÇÖZÜM SÜRESİ: ", hybrid_result_time)
QPU/CPU HIBRİT ÇÖZÜM SÜRESİ: 14.414310216903687
```

Şekil 7.3.2. Maximum Clique, Problemin QPU/CPU Hibrit ile Çözümlenmesi

Şekil 7.3.2 incelendiğinde, QPU/CPU hibrit çözümleyicinin probleme ~14.41 saniyede çözüm getirdiğini gözlemiyoruz.

QPU ile Çözümleme

```
In [347]: c_start = time.time()
gpu_result_a = dnx.maximum_clique(hymax, qpu_sampler)
c_end = time.time()
gpu_result_time = (c_end - c_start)
print("QPU ÇÖZÜM SÜRESİ: ", gpu_result_time)
QPU ÇÖZÜM SÜRESİ: 3.5077292919158936
```

Şekil 7.3.3. Maximum Clique, Problemin QPU ile Çözümlenmesi

Şekil 7.3.3 incelendiğinde ise, aynı problemin QPU tarafından çözümlenmesinin ~3.5 saniyede gerçekleştiğini görüyoruz. Bu durumda, bu problem özelinde QPU'nun QPU/CPU hibrit çözümleyiciye göre daha hızlı performans gösterdiğini söylemek yanlış olmayacağından emin oluyoruz.

CPU ile Çözümleme

```
In [349]: c_start = time.time()
cpu_result_a = dnx.maximum_clique(hymax, cpu_sampler)
c_end = time.time()
cpu_result_time = (c_end - c_start)
print("CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)
CPU ÇÖZÜM SÜRESİ: 65.40786409378052
```

Şekil 7.3.4. Maximum Clique, Problemin CPU ile Çözümlenmesi

Şekil 7.3.4'de ise aynı problemin CPU ile çözümlenmesinin ~65.41 saniye süredüğünü görebiliyoruz. Bu durumda QPU, CPU ve QPU/CPU hibrit çözümleyicisinin her ikisine kıyasla da daha hızlı performans göstermiştir.

Çözümlerin Karşılaştırılması

```
In [350]: print("QPU/CPU HİBRİT SONUÇ: ", hybrid_result_a)
print("QPU SONUÇ: ", gpu_result_a)
print("CPU SONUÇ: ", cpu_result_a)

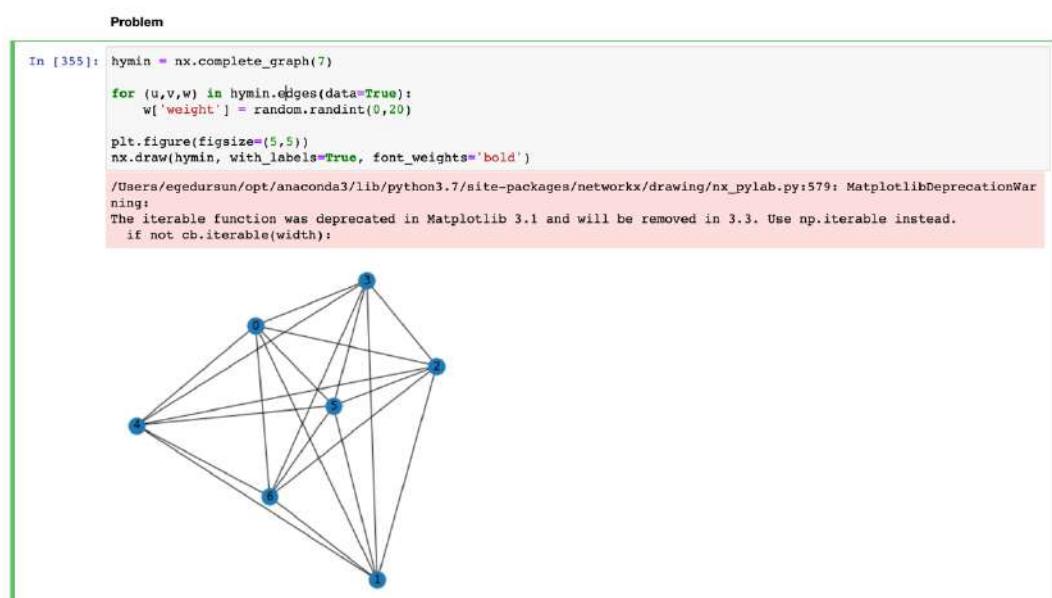
QPU/CPU HİBRİT SONUÇ: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24]
QPU SONUÇ: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24]
CPU SONUÇ: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24]
```

Şekil 7.3.5. Maximum Clique, QPU/CPU, QPU ve CPU Çözümlerinin Karşılaştırılması

Şekil 7.3.5'de QPU/CPU hibrit, QPU ve CPU çözümleyicilerin Maximum Clique problemleri için tanımlanan çizgeyi çözümlediklerinde ulaştıkları sonuçları gözlemliyoruz. Bu problem dahilinde ulaşılan sonuçların üç çözüm birimi için de ortak ve örtüşen nitelikte olduğunu söylemek yanlış olmayacağından emin oluyoruz. İleri bir araştırmada, farklı çizge örnekleri kullanılarak ulaşılan sonuçların tutarlılıkları daha etkin şekilde test edilebilir.

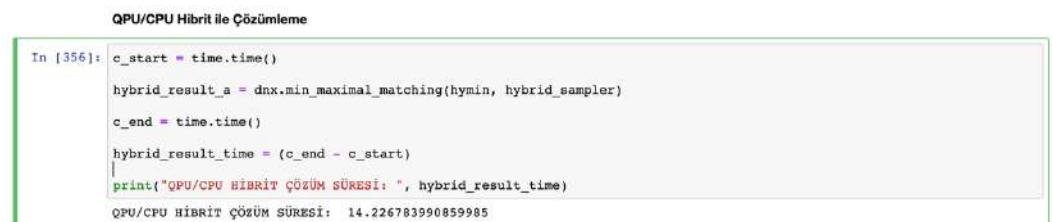
7.4 Minimum Maximal Matching Problemi

Bu bölümde Minimum Maximal Matching problemine örnek olacak şekilde Network-X kullanarak oluşturduğumuz çizgeyi D-Wave QPU/CPU hibrit, D-Wave QPU ve klasik CPU kullanarak çözümleyeceğiz. Ulaştığımız sonuçları ve performans metriklerini kaydedip raporlandıracagız.



Şekil 7.4.1. Minimum Maximal Matching, Problem

Şekil 7.4.1'de Minimum Maximal Matching problemine örnek olması amacıyla yarattığımız çizge görülmektedir. Bu çizge yedi adet birimden ve bu birimleri birbirine bağlayan yirmi bir adet bağlantından oluşmaktadır. Bu bağlantıların boyutları 0 ve 20 arasında rastgele olacak şekilde belirlenmiştir.



Şekil 7.4.2. Minimum Maximal Matching, Problemin QPU/CPU Hibrit ile Çözümlenmesi

Şekil 7.4.2'de problemin PQU/CPU hibriti ile çözümlenmesinin ~14.23 saniye sürdüğünü görmekteyiz.

```

QPU ile Çözümleme

In [357]: c_start = time.time()
           qpu_result_a = dnx.min_maximal_matching(hymin, qpu_sampler)
           c_end = time.time()
           qpu_result_time = (c_end - c_start)
           print("QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)
QPU ÇÖZÜM SÜRESİ:  5.954056978225708

```

Şekil 7.4.3. Minimum Maximal Matching, Problemin QPU ile Çözümlenmesi

Şekil 7.4.3'de ise aynı problemin QPU ile çözümlenmesinin ~5.95 saniye sürdüğünü görmekteyiz. Bu durumda, QPU'nun işlemi gerçekleştirme hızı QPU/CPU hibritine kıyasla fark edilir derecede daha fazladır.

```

CPU ile Çözümleme

In [358]: c_start = time.time()
           cpu_result_a = dnx.min_maximal_matching(hymin, cpu_sampler)
           c_end = time.time()
           cpu_result_time = (c_end - c_start)
           print("CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)
CPU ÇÖZÜM SÜRESİ:  7.748928070068359

```

Şekil 7.4.4. Minimum Maximal Matching, Problemin CPU ile Çözümlenmesi

Şekil 7.4.4'de son olarak problemin CPU ile çözümlenmesinin ~7.74 saniye sürdüğünü görüyoruz. Bu durumda QPU, CPU ve QPU/CPU hibritine kıyasla daha hızlı çözüm üretmiştir. Aynı şekilde CPU da QPU/CPU hibritine göre bu problem boyutu özelinde daha hızlı çözüm üretmeyi başarmıştır. Bu durumda mevcut problem için QPU/CPU hibritinin kullanılması fayda sağlamamaktadır.

Çözümlerin Karşılaştırılması

```
In [359]: print("QPU/CPU HİBRİT SONUÇ: ", hybrid_result_a)
print("QPU SONUÇ: ", gpu_result_a)
print("CPU SONUÇ: ", cpu_result_a)

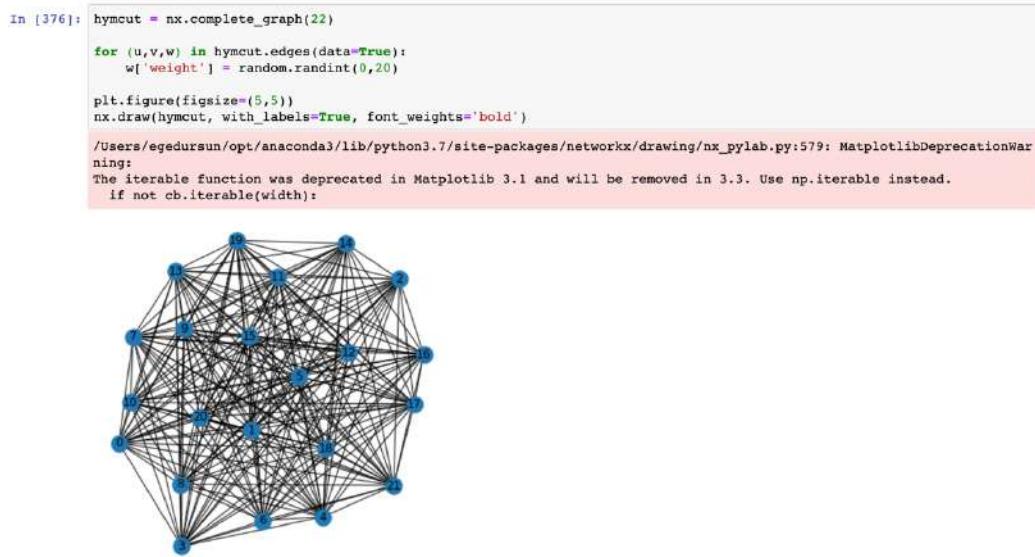
QPU/CPU HİBRİT SONUÇ: {(4, 5), (2, 6), (1, 3)}
QPU SONUÇ: {(1, 2), (4, 5), (1, 6), (2, 3), (0, 4), (2, 5), (3, 4)}
CPU SONUÇ: {(1, 5), (4, 6), (0, 2)}
```

Şekil 7.4.5 Minimum Maximal Matching, QPU/CPU Hibrit, QPU ve CPU Çözümlerinin Karşılaştırılması

Şekil 7.4.5'den anlaşılabileceği üzere Minimum Maximal Matching problemini örneklemek amacıyla oluşturulan çizgenin çözümleri QPU/CPU hibrit, QPU ve CPU arasında farklılık göstermekte ve tam olarak örtüşmemektedir.

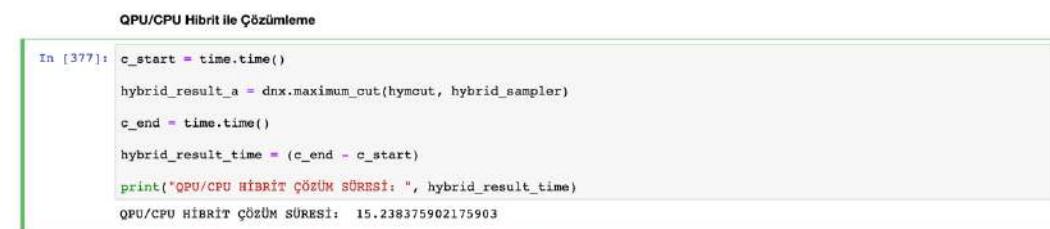
7.5 Maximum Cut Problemi

Bu bölümde Maximum Cut problemini örneklemek adına Network-X kütüphanesi kullanarak oluşturduğumuz örnek çizgeye D-Wave QPU/CPU Hibrit, D-Wave QPU ve CPU kullanarak çözüm getirdik. Elde ettiğimiz çözümleri ve performans metriklerini kaydederek raporladık.



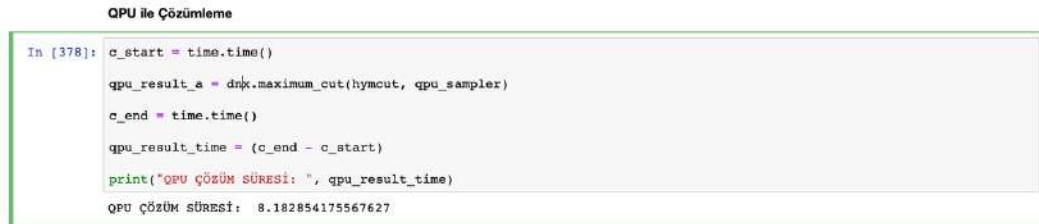
Şekil 7.5.1. Maximum Cut, Problem

Şekil 7.5.1'de Maximum Cut problemi için yarattığımız örnek çizgeyi görüyoruz. Bu çizge yirmi iki adet birimden ve bu birimleri birbirine bağlayan iki yüz otuz bir adet bağlantından oluşmaktadır. Bağlantıların ağırlıkları 0 ve 20 arasında rastgele olacak şekilde belirlenmiştir.



Şekil 7.5.2. Maximum Cut, QPU/CPU Hibrit ile Çözümlenmesi

Şekil 7.5.2'de tanımlanan Maximum Cut problemi örneğinin QPU/CPU hibriti tarafından çözümlenmesinin ~15.24 saniye sürdüğü gözlemlenmektedir.



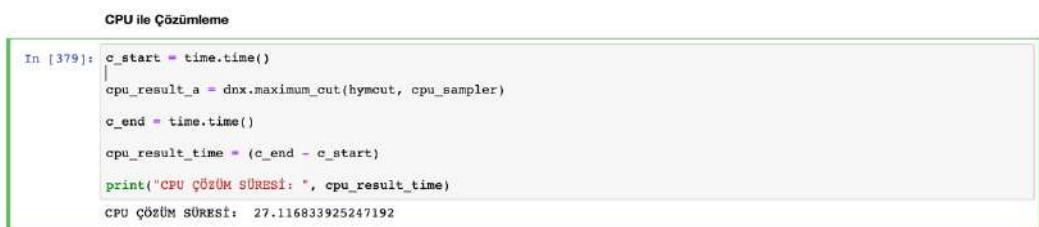
```

QPU ile Çözümleme
In [378]: c_start = time.time()
          qpu_result_a = dnx.maximum_cut(hymcut, qpu_sampler)
          c_end = time.time()
          qpu_result_time = (c_end - c_start)
          print("QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)
QPU ÇÖZÜM SÜRESİ:  8.182854175567627

```

Şekil 7.5.3. Maximum Cut, QPU ile Çözümleme

Şekil 7.5.3'de ise aynı problemin QPU ile çözümlenmesinin ~8.18 saniye süregünü gözlemliyoruz. Bu durumda QPU, QPU/CPU hibritine göre daha hızlı performans göstermiştir.



```

CPU ile Çözümleme
In [379]: c_start = time.time()
          cpu_result_a = dnx.maximum_cut(hymcut, cpu_sampler)
          c_end = time.time()
          cpu_result_time = (c_end - c_start)
          print("CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)
CPU ÇÖZÜM SÜRESİ:  27.116833925247192

```

Şekil 7.5.4. Maximum Cut, CPU ile Çözümleme

Şekil 7.5.4'de son olarak aynı problemin CPU ile çözümlenmesinin ~27.11 saniye süregünü gözlemlemekteyiz. Bu durumda hem QPU/CPU hibriti, hem de QPU, mevcut problem özelinde CPU'ya kıyasla daha iyi performans göstermiştir. QPU ise, QPU/CPU hibritine göre daha iyi performans göstermiştir. Bu durumda problemi çözümlemek için QPU çözümleyici kullanmanın en hızlı yol olduğu söylenebilir.

Çözümlerin Karşılaştırılması

```
In [380]: print("QPU/CPU HİBRİT SONUÇ: ", hybrid_result_a)
print("QPU SONUÇ: ", qpu_result_a)
print("CPU SONUÇ: ", cpu_result_a)

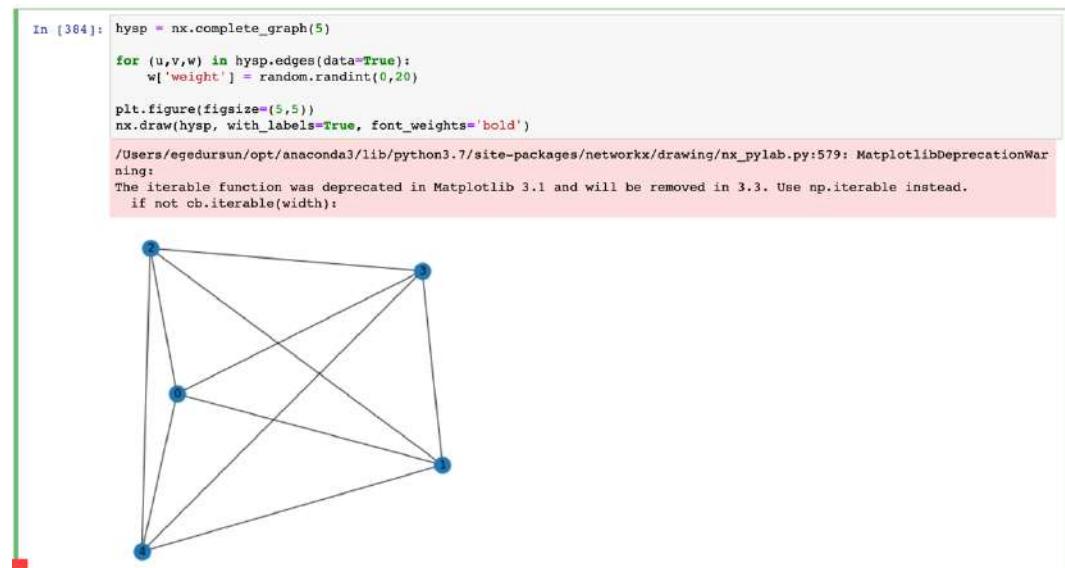
QPU/CPU HİBRİT SONUÇ: {4, 5, 7, 8, 9, 10, 13, 14, 15, 16, 21}
QPU SONUÇ: {5, 7, 9, 11, 12, 13, 15, 17, 18, 20, 21}
CPU SONUÇ: {1, 2, 5, 6, 8, 10, 11, 12, 16, 17, 20}
```

Şekil 7.5.5. Maximum Cut, QPU/CPU Hibrit, QPU ve CPU Çözümlerinin Karşılaştırılması

Şekil 7.5.5’de Maximum Cut problemi için oluşturulan örneğin QPU/CPU hibrit, QPU ve CPU çözümleyiciler tarafından çözümlendiğinde ulaşılan sonuçları görüyoruz. Anlaşılacağı üzere, bu problemde de QPU/CPU hibrit, QPU ve CPU’nun elde ettikleri çözümler tam anlamıyla örtüşmemektedir. Örtüşmeme sebeplerinin araştırılması için ileri çalışmalar önerilir.

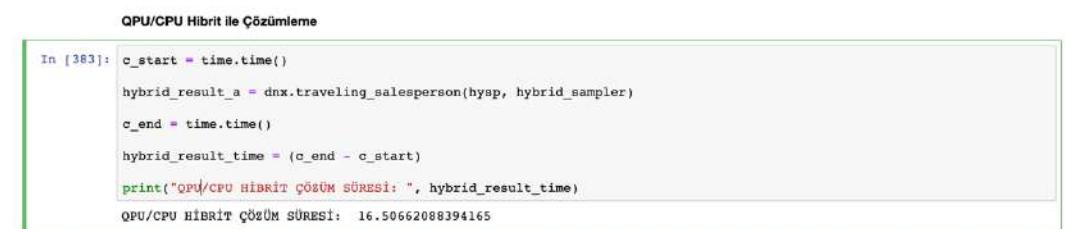
7.6 Traveling Salesperson Problemi

Bu bölümde Traveling Salesperson problemine örnek olacak şekilde Network-X kütüphanesi kullanılarak oluşturduğumuz çizgeye D-Wave QPU/CPU hibrit, D-Wave QPU ve klasik CPU kullanarak çözümler ürettiğimiz Elde ettiğimiz sonuçları kaydettik ve raporlandırdık.



Şekil 7.6.1. Traveling Salesperson, Problem

Şekil 7.6.1'de Traveling Salesperson problemi için örnek olarak oluşturduğumuz çizge yer almaktadır. Bu çizge beş adet birimden ve bu birimleri birbirine bağlayan on adet bağlantı noktasıından meydana gelmektedir. Bağlantıların ağırlıkları 0 ve 20 arasında rastgele olacak şekilde belirlenmiştir.



Şekil 7.6.2. Traveling Salesperson, Problemin QPU/CPU Hibrit ile Çözümlenmesi

Şekil 7.6.2'de Traveling Salesperson problemini örnekleme amacıyla oluşturduğumuz çizgenin QPU/CPU hibrit çözümleyici ile çözülmesinin ~16.51 saniye sürdüğünü görmekteyiz.

```
QPU ile Çözümleme
In [385]: c_start = time.time()
qpu_result_a = dnx.traveling_salesperson(hysp, qpu_sampler)
c_end = time.time()
qpu_result_time = (c_end - c_start)
print("QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)
QPU ÇÖZÜM SÜRESİ:  9.78623914718628
```

Şekil 7.6.3. Traveling Salesperson, Problemin QPU ile Çözümlenmesi

Şekil 7.6.3'de ise aynı problemin QPU ile çözümlenmesinin ~9.79 saniye sürdüğünü görüyoruz. Bu durumda mevcut problemde QPU'nun QPU/CPU hibritine kıyasla daha hızlı çözümleme sağladığını söyleyebiliriz.

```
CPU ile Çözümleme
In [386]: c_start = time.time()
cpu_result_a = dnx.traveling_salesperson(hysp, cpu_sampler)
c_end = time.time()
cpu_result_time = (c_end - c_start)
print("CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)
CPU ÇÖZÜM SÜRESİ:  185.0526249408722
```

Şekil 7.6.4. Traveling Salesperson, Problemin CPU ile Çözümlenmesi

Şekil 7.6.4'de son olarak problemin CPU ile çözümlenmesinin ~185.05 saniye sürdüğünü görüyoruz. Bu sonuçlar ışığında, mevcut problemin çözümlenmesi açısından en avantajlı çözümleyicinin QPU, ikinci en avantajlı çözümleyicinin QPU/CPU hibriti olduğunu; sonuca ulaşma konusunda en yavaş kalan çözümleyicinin ise CPU olduğunu fark edebiliriz.

Çözümlerin Karşılaştırılması

```
In [387]: print("QPU/CPU HİBRİT SONUÇ: ", hybrid_result_a)
print("QPU SONUÇ: ", qpu_result_a)
print("CPU SONUÇ: ", cpu_result_a)

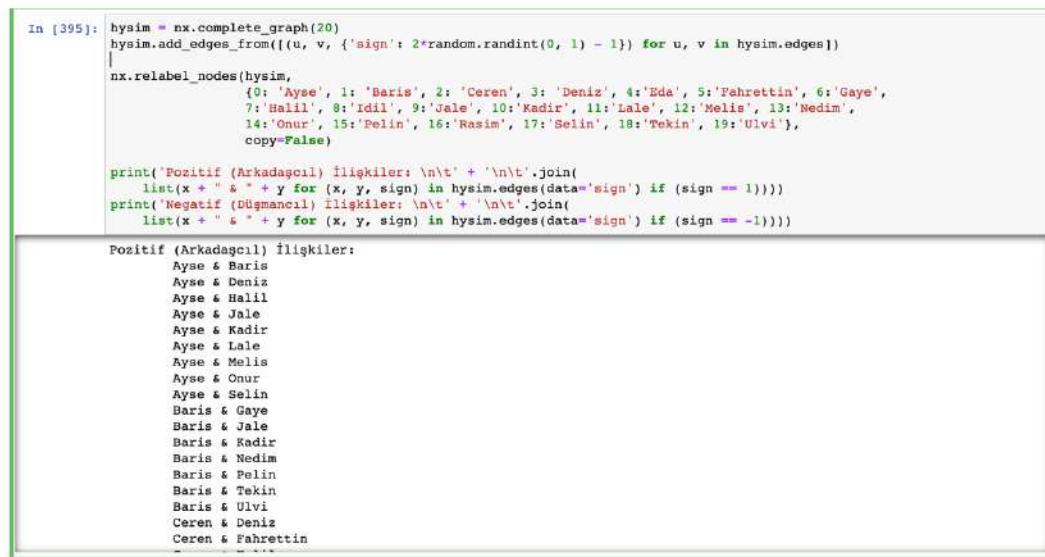
QPU/CPU HİBRİT SONUÇ: [2, 9, 5, 7, 6, 0, 1, 3, 8, 4]
QPU SONUÇ: [3, 2, 3, 1, 4]
CPU SONUÇ: [2, 1, 0, 3, 4]
```

Şekil 7.6.5. Traveling Salesperson, QPU/CPU Hibrit, QPU ve CPU Sonuçlarının Karşılaştırılması

Şekil 7.6.5 incelendiğinde, özellikle QPU ve CPU'nun sonuçlarıyla QPU/CPU hibritinin sonuçlarının örtüşmezlik taşıdığı göze çarpmaktadır. QPU ve CPU'nun çözümleri de kendi arasında daha az miktarda örtüşmezlikler taşımakta olup; çözümleyicilerin sonuçlarının tutarsızlık sebepleri araştırılmalıdır.

7.7 Structural Imbalance Problemi

Bu bölümde Structural Imbalance problemine örnek olarak Network-X kütüphanesi kullanarak yarattığımız örnek probleme D-Wave QPU/CPU Hibriti, D-Wave QPU ve CPU kullanarak çözüm üretmeye çalıştık. Ulaştığımız çözümleri ve performans metriklerini kaydettik ve raporlandırdık.



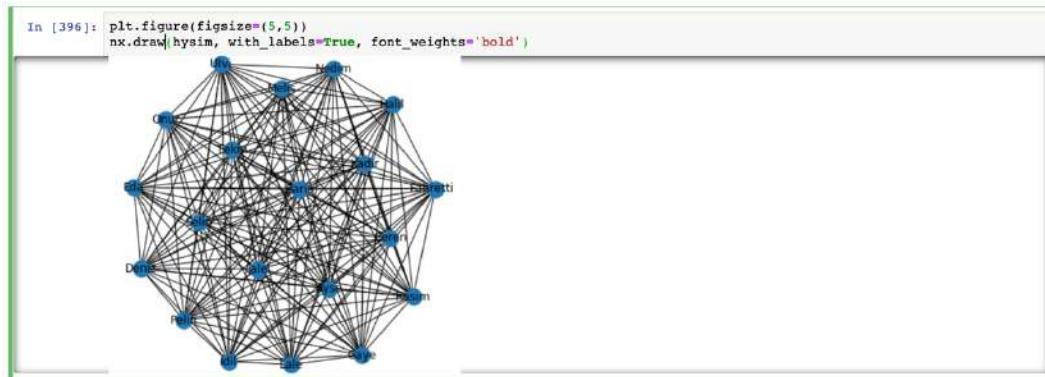
```
In [395]: hysim = nx.complete_graph(20)
hysim.add_edges_from([(u, v, {'sign': 2*random.randint(0, 1) - 1}) for u, v in hysim.edges])
nx.relabel_nodes(hysim,
                  {0: 'Ayse', 1: 'Baris', 2: 'Ceren', 3: 'Deniz', 4: 'Eda', 5: 'Fahrettin', 6: 'Gaye',
                   7: 'Halil', 8: 'Idil', 9: 'Jale', 10: 'Kadir', 11: 'Lale', 12: 'Melis', 13: 'Nedim',
                   14: 'Onur', 15: 'Pelin', 16: 'Rasim', 17: 'Selin', 18: 'Tekin', 19: 'Ulvi'},
                  copy=False)

print('Positif (Arkadaşcil) ilişkiler: \n\t' + '\n\t'.join(
    list(x + " & " + y for (x, y, sign) in hysim.edges(data='sign') if (sign == 1))))
print('Negatif (Düşmancil) ilişkiler: \n\t' + '\n\t'.join(
    list(x + " & " + y for (x, y, sign) in hysim.edges(data='sign') if (sign == -1))))
```

```
Positif (Arkadaşcil) ilişkiler:
Ayse & Baris
Ayse & Deniz
Ayse & Halil
Ayse & Jale
Ayse & Kadir
Ayse & Lale
Ayse & Melis
Ayse & Onur
Ayse & Selin
Baris & Gaye
Baris & Jale
Baris & Kadir
Baris & Nedim
Baris & Pelin
Baris & Tekin
Baris & Ulvi
Ceren & Deniz
Ceren & Fahrettin
-- -- --
```

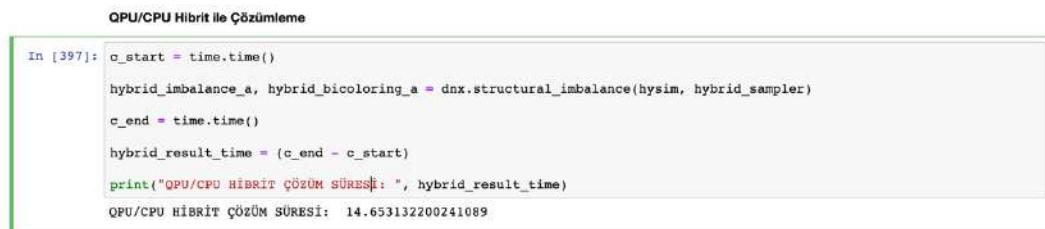
Şekil 7.7.1. Structural Imbalance, Problem

Şekil 7.7.1'de tanımladığımız problemi görüyoruz. Problemin tanımında kullanılan çizge, yirmi adet birim ve bu birimleri bağlayan yüz doksan adet bağlantıdan oluşmaktadır. Bu bağlantıların her biri birimlerin birbiri ile olan pozitif/negatif ilişkilerini ifade etmekte olmakta ve rastgele olarak belirlenmektedir. Birimlere örnek isimlendirmeler yapılmıştır ve ekranda görüntülenmiştir. Daha detaylı açıklamalar Ek1'de incelenebilir.



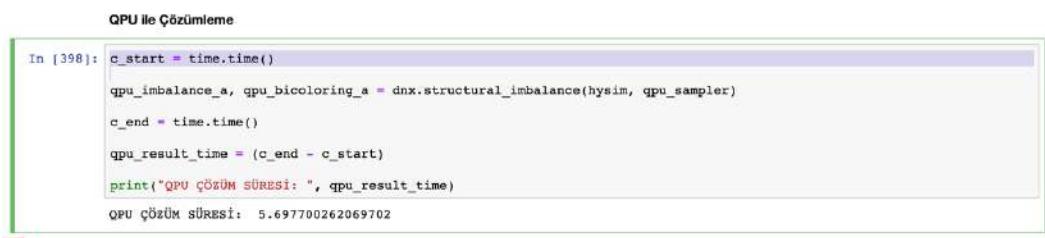
Şekil 7.7.2. Structural Imbalance, Problem Çizge

Şekil 7.7.2'de Structural Imbalance problemine örnek olarak tasarlanan çizgenin görüntüsünü inceleyebiliriz.



Şekil 7.7.3. Structural Imbalance, Problemin QPU/CPU Hibrit ile Çözümlenmesi

Şekil 7.7.3'de problemin QPU/CPU hibriti ile çözümlenmesinin ~14.65 saniye süregünü gözlemlemekteyiz.



Şekil 7.7.4. Structural Imbalance, Problemin QPU ile Çözümlenmesi

Şekil 7.7.4'de problemin QPU ile çözümlenmesinin ise ~5.7 saniye süregünü görüyoruz. Bu durumda QPU, QPU/CPU hibriti ile kıyaslandığında probleme daha hızlı çözüm bulabilmektedir.

CPU ile Çözümleme

```
In [399]: c_start = time.time()

cpu_imbalance_a, cpu_bicoloring_a = dnx.structural_imbalance(hysim, cpu_sampler)

c_end = time.time()

cpu_result_time = (c_end - c_start)

print("CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)
CPU ÇÖZÜM SÜRESİ: 3.2858550548553467
```

Şekil 7.7.5. Structural Imbalance, Problemin CPU ile Çözümlenmesi

Şekil 7.7.5’de problemin CPU ile çözümlenmesinin ise ~3.28 saniye sürdüğünü görüyoruz. Bu durumda mevcut problem için CPU, hem QPU/CPU hibriti hem de QPU ya göre daha hızlı bir şekilde probleme çözüm getirebilmiştir. CPU’nun işlem hızının azaldığı noktayı yakalayabilmek için daha büyük bir problem boyutu kullanmak denenebilir.

Çözümlerin Karşılaştırılması

```
In [401]: for edge in hysim.edges:
    hysim.edges[edge]['frustrated'] = edge in hybrid_imbalance_a
for node in hysim.nodes:
    hysim.nodes[node]['color'] = hybrid_bicoloring_a[node]

print("PROBLEM (QPU/CPU):")
print('GRUP A: \n\t' + '\n\t'.join(
    list(person for (person, color) in hybrid_bicoloring_a.items() if (color == 0))))
print('GRUP B: \n\t' + '\n\t'.join(
    list(person for (person, color) in hybrid_bicoloring_a.items() if (color == 1)))
print('DENGESİZ/INSTABİL İLİŞKİLER: \n\t' + '\n\t'.join(
    list(x + " & " + y for (x, y) in hybrid_imbalance_a.keys())))

PROBLEM (QPU/CPU):
GRUP A:
    Baris
    Gaye
    Idil
    Kadir
    Lale
    Nedim
    Tekin
GRUP B:
    Ayse
    Ceren
    Deniz
    Eda
    Fahrettin
    Halil
    Jale
    Melis
    Onur
    Pelin
    Rasim
    Selin
    Ulvi
DENGESİZ/INSTABİL İLİŞKİLER:
    Ayse & Baris
    Ayse & Ceren
    Ayse & Eda
    Ayse & Fahrettin
    Ayse & Kadir
```

Şekil 7.7.6. Structural Imbalance, QPU/CPU Çözümleri

Şekil 7.7.6’da probleme QPU/CPU hibriti aracılığıyla ulaşılan çözümleri gözlemleyebiliriz.

```
In [403]: for edge in hysim.edges:
    hysim.edges[edge]['frustrated'] = edge in gpu_imbalance_a
for node in hysim.nodes:
    hysim.nodes[node]['color'] = gpu_bicoloring_a[node]

print("PROBLEM (QPU):")
print('GRUP A: \n\t'.join(list(person for (person, color) in gpu_bicoloring_a.items() if (color == 0))))
print('GRUP B: \n\t'.join(list(person for (person, color) in gpu_bicoloring_a.items() if (color == 1))))
print('DENGESİZ/INSTABİL İLİŞKİLER: \n\t'.join(list(x + " & " + y for (x, y) in gpu_imbalance_a.keys())))

PROBLEM (CPU):
GRUP A:
    Eda
    Halil
    Idil
    Kadir
    Lale
    Onur
    Pelin
    Rasim

GRUP B:
    Ayse
    Baris
    Ceren
    Deniz
    Fahrettin
    Gaye
    Jale
    Melis
    Nedim
    Selin
    Tekin
    Ulvi

DENGESİZ/INSTABİL İLİŞKİLER:
    Ayse & Ceren
    Ayse & Fahrettin
    Ayse & Gaye
    Ayse & Halil
    Ayse & Kadir
    Ayse & Lale
    Ayse & Nedim
    Ayse & Onur
    Ayse & Tekin
    Ayse & Ulvi
```

Şekil 7.7.7. Structural Imbalance, QPU Çözümleri

Şekil 7.7.7'de probleme QPU aracılığıyla ulaşılan çözümleri gözlemlayabiliriz.

```
In [404]: for edge in hysim.edges:
    hysim.edges[edge]['frustrated'] = edge in cpu_imbalance_a
for node in hysim.nodes:
    hysim.nodes[node]['color'] = cpu_bicoloring_a[node]

print("PROBLEM (CPU):")
print('GRUP A: \n\t'.join(list(person for (person, color) in cpu_bicoloring_a.items() if (color == 0))))
print('GRUP B: \n\t'.join(list(person for (person, color) in cpu_bicoloring_a.items() if (color == 1))))
print('DENGESİZ/INSTABİL İLİŞKİLER: \n\t'.join(list(x + " & " + y for (x, y) in cpu_imbalance_a.keys())))

PROBLEM (CPU):
GRUP A:
    Ayse
    Baris
    Gaye
    Idil
    Kadir
    Lale
    Nedim

GRUP B:
    Ceren
    Deniz
    Eda
    Fahrettin
    Halil
    Jale
    Melis
    Onur
    Pelin
    Rasim
    Selin
    Tekin
    Ulvi

DENGESİZ/INSTABİL İLİŞKİLER:
    Ayse & Deniz
    Ayse & Gaye
    Ayse & Halil
    Ayse & Idil
    Ayse & Jale
    Ayse & Melis
```

Şekil 7.7.8. Structural Imbalance, CPU Çözümleri

Şekil 7.7.8'de probleme CPU aracılığıyla ulaşılan çözümleri gözlemleyebiliriz. Her üç çözümleyicinin çözüm kümeleri de incelendiğinde, QPU/CPU hibriti, QPU ve CPU ile ulaşılan çözümler karşılaştırıldığında tam olarak birbirleri ile örtüşmediğini söyleyebiliriz. Bu örtüşmezliğin sebepleri üzerine ileri araştırma yapılması doğru olacaktır.

8. YAPISAL DENGESİZLİK PROBLEMLERİNİN ÇÖZÜMLENMESİ

Bu bölümde Yapısal Dengesizlik problemine ilişkin örnek üç adet problem tanımladık ve bu problemleri D-Wave QPU/CPU hibriti, D-Wave QPU ve klasik CPU kullanarak çözümledik. Problemlerimizi tanımlarken Network-X kütüphanesinden faydalandık. Çözümleyicilerin elde ettiği sonuçları kaydettik ve raporlandırdık.

Çalışmamızın bu bölümünde öncelikli olarak Network-X kütüphanesi kullanarak oluşturduğumuz problemleri sırayla tanımlayacağız. Ardından, takip eden başlıklarda sırasıyla bu problemlerin QPU/CPU hibriti, QPU ve CPU ile çözümlenmesini sağlayacak; problemleri çözme sürelerini ölçümleyeceğiz. Sonrasında, her üç problem için ulaşılan çözümleri karşılaştıracağız.

8.1 Örnek Yapısal Dengesizlik Problemlerinin Tanımlanması

Bu bölümde çalışmamız dahilinde çözümleyeceğimiz örnek yapısal dengesizlik problemlerini Network-X kütüphanesi kullanarak tanımlayacak ve tanımladığımız problemleri oluşturan detaylardan bahsedeceğiz.

1. Problem

```
In [582]: sil = nx.complete_graph(35)
sil.add_edges_from([(u, v, {'sign': 2*random.randint(0, 1) - 1}) for u, v in sil.edges])
people = {}

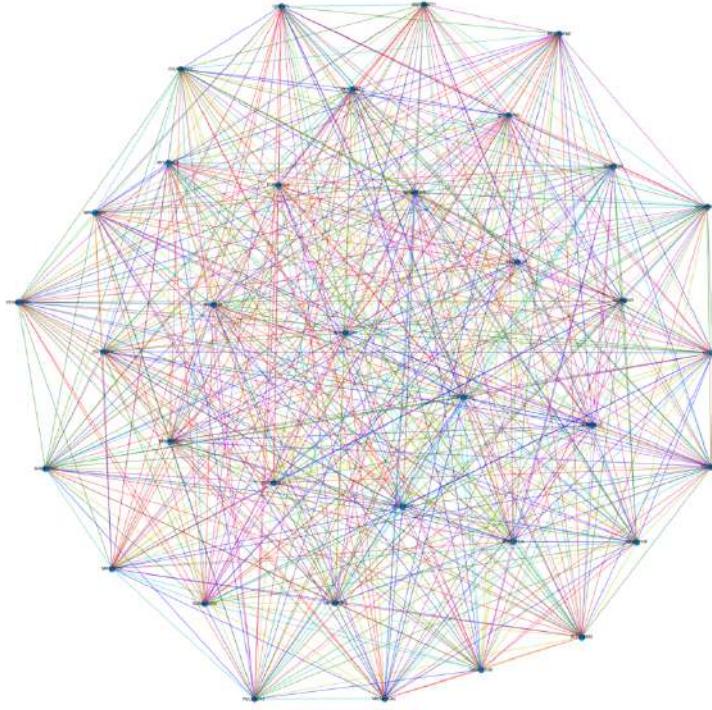
for i in range(0, 35):
    people[i] = ''.join(random.choice(string.ascii_uppercase) for _ in range(8))

nx.relabel_nodes(sil,
                 people,
                 copy=False)

#print('Positif (Arkadaşılık) ilişkiler: \n\t' + '\n\t'.join(list(x + " & " + y for (x, y, sign) in sil.edges(data='sign'))))
#print('Negatif (Düşmanlık) ilişkiler: \n\t' + '\n\t'.join(list(x + " & " + y for (x, y, sign) in sil.edges(data='sign'))))
```

Şekil 8.1.1. Yapısal Dengesizlik Problemi,, 1. Problem

Şekil 8.1.1'de çalışmamız dahilinde tanımladığımız ilk yapısal dengesizlik problemini gözlemliyoruz. Bu problem, otuz beş adet birimden ve bu birimleri birbirine bağlayan beş yüz doksan beş adet bağlantıdan oluşmaktadır. Bu bağlantılar çizgeye dahil olan birimlerin/kısilerin birbirleri ile olan ilişkilerini temsile etmekte olup pozitif veya negatif olarak rastgele değer almaktadır. Çizgede yer alan her birime rastgele olacak şekilde isimlendirme yapılmıştır.



Şekil 8.1.2. Yapısal Dengesizlik Problemi, 1. Problem Çizge

Şekil 8.1.2'de ürettiğimiz çizgenin ekranda oluşturulan temsilini görmekteyiz. Bu görselin oluşturulmasında matplotlib ve Network-X kütüphanelerinden faydalanılmıştır. Çizgeyi oluşturan birimler ve birimleri birbirine bağlayan bağlantılar şekilde açıkça görülebilir.

2. Problem

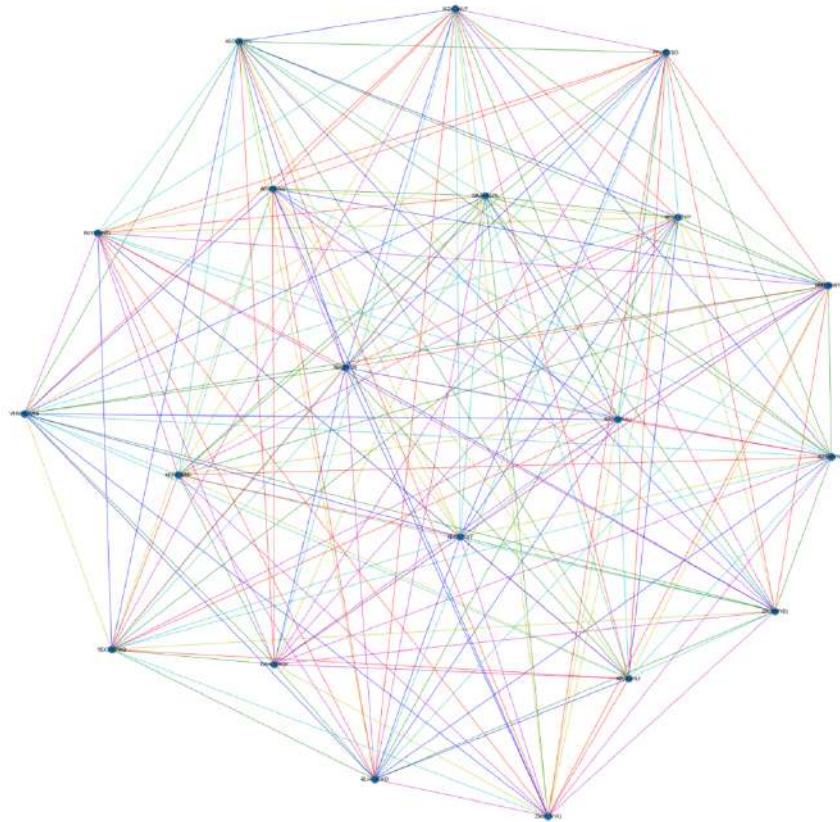
```
In [578]: s12 = nx.complete_graph(20)
s12.add_edges_from([(u, v, {'sign': 2*random.randint(0, 1) - 1}) for u, v in s12.edges])
people = {}
for i in range(0, 20):
    people[i] = ''.join(random.choice(string.ascii_uppercase) for _ in range(8))
nx.relabel_nodes(s12,
                 people,
                 copy=False)

#print('Positif (Arkadaşçı) ilişkiler: \n' + '\n'.join(list(x + " & " + y for (x, y, sign) in sil.edges(data='sign'))))
#print('Negatif (Düşmancı) ilişkiler: \n' + '\n'.join(list(x + " & " + y for (x, y, sign) in sil.edges(data='sign'))))
```

Şekil 8.1.3. Yapısal Dengesizlik Problemi, 2. Problem

Şekil 8.1.3'de oluşturduğumuz ikinci yapısal dengesizlik probleminin tanımını gözlemleyebiliriz. Bu çizge yirmi adet birimden ve bu birimlerin birbirine bağlayan yüz doksan adet bağlantı noktasından oluşmaktadır. Bu bağlantı noktaları önceki problemde olduğu gibi birimlerin birbirleri ile olan ilişkilerini temsil etmekte ve ilişkinin iyi ya da kötü olmasını ifade edecek şekilde pozitif veya negatif

değer almaktadır. Değerler rastgele olacak şekilde oluşturulmaktadır. Çizgede yer alan birimlere de rastgele olacak isimler verilmektedir.



Şekil 8.1.4. Yapısal Dengesizlik Problemi, 2. Problem Çizge

Şekil 8.1.4'de ürettiğimiz ikinci yapısal dengesizlik probleminin görsel temsilini görmekteyiz. Bu görselleştirme yaratılırken matplotlib ve Network-X kütüphanelerinden faydalanilmıştır.

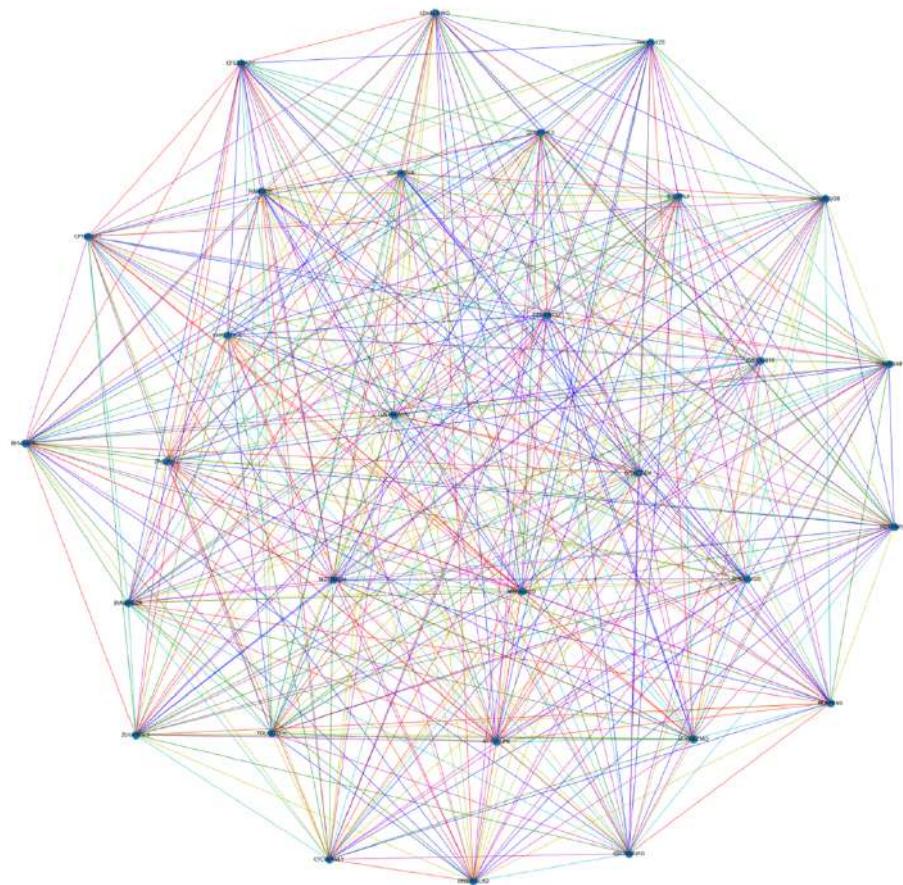
3. Problem

```
In [580]: s13 = nx.complete_graph(30)
s13.add_edges_from([(u, v, {'sign': 2*random.randint(0, 1) - 1}) for u, v in s13.edges])
people = {}
for i in range(0, 30):
    people[i] = ''.join(random.choice(string.ascii_uppercase) for _ in range(8))
nx.relabel_nodes(s13,
                 people,
                 copy=False)

#print('Pozitif (Arka dasıl) İlighiler: \n\t' + '\n\t'.join(list(x + " & " + y for (x, y, sign) in sil.edges(data='sign')))
#print('Negatif (Dügmancıl) İlighiler: \n\t' + '\n\t'.join(list(x + " & " + y for (x, y, sign) in sil.edges(data='sign'))))
```

Şekil 8.1.5. Yapısal Dengesizlik Problemi, 3. Problem

Şekil 8.1.5'de ürettiğimiz üçüncü örnek yapısal dengesizlik probleminin tanımını gözlemliyoruz. Bu çizge toplamda otuz birimden ve bu birimleri birbirine bağlayan dört yüz otuz beş adet bağlantı noktasından oluşmaktadır. Bağlantı noktaları pozitif ve negatif ilişkileri temsil edecek şekilde rastgele 1 veya -1 değeri almaktadır. Birimler için rastgele isimlendirmeler oluşturulmuştur.



Şekil 8.1.6. Yapısal Dengesizlik Problemi, 3.Problem Çizge

Şekil 8.1.6'da oluşturduğumuz üçüncü probleme ait çizgeyi görmekteyiz. Bu çizge de aynı şekilde matplotlib ve Network-X kütüphaneleri kullanılarak oluşturulmuştur.

8.2 Problemlerin CPU Kullanılarak Çözümlenebilirliği

Bu bölümde üretilen problemler, klasik CPU tarafından çözümlenmesi çok uzun sürecek boyuta sahip olduğundan, ölçümlemeler sadece QPU/CPU hibriti ve QPU ile gerçekleştirilmiştir. Daha güçlü bir CPU kullanılarak, bu bölümde çözümlenen problemlere daha kabul edilebilir zamanlarda çözümler getirilebilir. Ancak bu defa, sözü edilen işlemcinin sınırlarını aşan yeni bir problem boyutu olacaktır.

Devam eden başlıklarda, ürettiğimiz üç adet problemin QPU/CPU hibriti ve QPU ile çözümlenmesini sağladık ve ulaştığımız sonuçları raporlandırdık. Bu bölümün temel amacı, klasik CPU'larca çözümlenmesi oldukça güç olan problemlerin QPU/CPU ve QPU kullanarak çözümlenebilmesinin artık mümkün olup olmadığını test etmektir. Çalışmamız sayesinde bu konu hakkında ileri araştırmalar yapılmasını teşvik etmeyi amaçlıyoruz.

8.3 Problemlerin D-Wave QPU Kullanılarak Çözümlenmesi

Bu bölümde, önceki başlıklarda tanımlanan örnek yapısal dengesizlik problemlerinin QPU ile çözümlenmesini sağladık. Ulaştığımız sonuçları kaydettik ve raporlandırdık.

```
Problem 1
In [584]: c_start = time.time()
qpu_imbalance_a, qpu_bicoloring_a = dnx.structural_imbalance(si1, qpu_sampler)
c_end = time.time()
qpu_result_time = (c_end - c_start)
print("1. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)
1. PROBLEM - QPU ÇÖZÜM SÜRESİ:  11.170470237731934
```

Şekil 8.3.1. Yapısal Dengesizlik Problemi, 1.Problemin QPU ile Çözümlenmesi

Şekil 8.3.1'de gözlemlenebileceği üzere oluşturulan birinci problemin QPU ile çözümlenmesi ~11.17 saniye sürmüştür.

```
Problem 2
In [587]: c_start = time.time()
qpu_imbalance_b, qpu_bicoloring_b = dnx.structural_imbalance(si2, qpu_sampler)
c_end = time.time()
qpu_result_time = (c_end - c_start)
print("2. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)
2. PROBLEM - QPU ÇÖZÜM SÜRESİ:  5.7992589473724365
```

Şekil 8.3.2. Yapısal Dengesizlik Problemi, 2.Problemin QPU ile Çözümlenmesi

Şekil 8.3.2'de ise ikinci problemin QPU ile çözümlenmesinin ~5.8 saniyede gerçekleştiğini görüyoruz.

```
Problem 3
In [588]: c_start = time.time()
qpu_imbalance_c, qpu_bicoloring_c = dnx.structural_imbalance(si3, qpu_sampler)
c_end = time.time()
qpu_result_time = (c_end - c_start)
print("3. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)
3. PROBLEM - QPU ÇÖZÜM SÜRESİ:  7.823553085327148
```

Şekil 8.3.3. Yapısal Dengesizlik Problemi, 3.Problemin QPU ile Çözümlenmesi

Son olarak Şekil 8.3.3'de üçüncü problemin QPU ile çözümlenmesinin ~7.82 saniye sürdüğünü gözlemliyoruz.

8.4 Problemlerin D-Wave QPU/CPU Hibritleme Kullanılarak Çözümlenmesi

Bu bölümde, önceki başlıklarda tanımlanan örnek yapısal dengesizlik problemlerinin QPU/CPU hibriti ile çözümlenmesini sağladık. Ulaştığımız sonuçları kaydettik ve raporlandırdık.

```
In [589]: c_start = time.time()
hybrid_imbalance_a, hybrid_bicoloring_a = dnx.structural_imbalance(si1, hybrid_sampler)
c_end = time.time()
hybrid_result_time = (c_end - c_start)
print("1. PROBLEM - QPU/CPU HİBRİT ÇÖZÜM SÜRESİ: ", hybrid_result_time)
1. PROBLEM - QPU/CPU HİBRİT ÇÖZÜM SÜRESİ:  11.206799030303955
```

Şekil 8.4.1. Yapısal Dengesizlik Problemi, 1. Problemin QPU/CPU Hibriti ile Çözümlenmesi

Şekil 8.4.1'de birinci problemin QPU/CPU hibriti ile çözümlenmesinin ~11.21 saniye sürdüğü gözlemlenmektedir.

```
In [590]: c_start = time.time()
hybrid_imbalance_b, hybrid_bicoloring_b = dnx.structural_imbalance(si2, hybrid_sampler)
c_end = time.time()
hybrid_result_time = (c_end - c_start)
print("2. PROBLEM - QPU/CPU HİBRİT ÇÖZÜM SÜRESİ: ", hybrid_result_time)
2. PROBLEM - QPU/CPU HİBRİT ÇÖZÜM SÜRESİ:  15.674062252044678
```

Şekil 8.4.2. Yapısal Dengesizlik Problemi, 2. Problemin QPU/CPU Hibriti ile Çözümlenmesi

Şekil 8.4.2'de ikinci problemin QPU/CPU hibriti ile çözümlenmesinin ~15.67 saniye sürdüğü gözlemlenmektedir.

```
In [591]: c_start = time.time()
          hybrid_imbalance_c, hybrid_bicoloring_c = dnx.structural_imbalance(si3, hybrid_sampler)
          c_end = time.time()
          hybrid_result_time = (c_end - c_start)
          print("3. PROBLEM - QPU/CPU HİBRİT ÇÖZÜM SÜRESİ: ", hybrid_result_time)
3. PROBLEM - QPU/CPU HİBRİT ÇÖZÜM SÜRESİ: 16.554741144180298
```

Şekil 8.4.3. Yapısal Dengesizlik Problemi, 3. Problemin QPU/CPU Hibriti ile Çözümlenmesi

Şekil 8.4.3’de son olarak üçüncü problemin QPU/CPU hibriti ile çözümlenmesinin ~16.55 saniye sürdüğü görülmektedir.

Çözümleme süreleri dikkate alındığında, birinci problem dahilinde QPU ve QPU/CPU hibritinin birbirine çok yakın performanslara sahip olduğunu söyleyebiliriz. İkinci ve üçüncü problemi incelediğimizde ise, QPU’nun QPU/CPU hibritine kıyasla daha hızlı çözümleme sağladığını görmekteyiz.

8.5 Sonuçların Karşılaştırılması

Bu bölümde çalışmamız dahilinde ürettiğimiz yapısal dengesizlik problemlerine ilişkin QPU/CPU hibriti ve QPU tarafından getirilen çözümleri birbirleri ile karşılaştırdık ve yorumladık.

```

1. Problem

In [592]: for edge in sil.edges:
    sil.edges[edge]['frustrated'] = edge in gpu_imbalance_a
for node in sil.nodes:
    sil.nodes[node]['color'] = gpu_bicoloring_a[node]

print("1. PROBLEM (QPU):")
print('GRUP A: \n' + '\n'.join(list(person for (person, color) in gpu_bicoloring_a.items() if (color == 0))))
print('GRUP B: \n' + '\n'.join(list(person for (person, color) in gpu_bicoloring_a.items() if (color == 1))))
print('DENGESİZ/INSTABİL İLİŞKİLER: \n' + '\n'.join(list(x + " & " + y for (x, y) in gpu_imbalance_a.keys())))

```

VCCXJHD
DENGESİZ/INSTABİL İLİŞKİLER:
MOYXRFBP & AXWXFQO
MOYXRFBP & SUBJWZOO
MOYXRFBP & ESDIHDJH
MOYXRFBP & BRRLBSVY
MOYXRFBP & LJHJOJYL
MOYXRFBP & AWFABBTH
MOYXRFBP & MKVTOATV
MOYXRFBP & EDKCLCLLA
MOYXRFBP & TEUQICUF
MOYXRFBP & GRGUWBGJ
MOYXRFBP & OCHADGYG
MOYXRFBP & JSEVUJHH
MOYXRFBP & VAGLFUOB
MOYXRFBP & FHUGNOEV
MOYXRFBP & FUPSMTGE
MOYXRFBP & KIVVGDHT
MOYXRFBP & MNSAGEAC

Şekil 8.5.1. Yapısal Dengesizlik Problemi, 1. Problem QPU Çözümleri

Şekil 8.5.1'de birinci probleme QPU çözümleyici tarafından üretilen sonuçları gözlemlayabiliriz. Sonuçlar daha detaylı olarak Ek 1'de belirtilmiştir.

```

In [595]: for edge in sil.edges:
    sil.edges[edge]['frustrated'] = edge in hybrid_imbalance_a
for node in sil.nodes:
    sil.nodes[node]['color'] = hybrid_bicoloring_a[node]

print("1. PROBLEM (QPU/CPU):")
print('GRUP A: \n' + '\n'.join(list(person for (person, color) in hybrid_bicoloring_a.items() if (color == 0))))
print('GRUP B: \n' + '\n'.join(list(person for (person, color) in hybrid_bicoloring_a.items() if (color == 1))))
print('DENGESİZ/INSTABİL İLİŞKİLER: \n' + '\n'.join(list(x + " & " + y for (x, y) in hybrid_imbalance_a.keys())))

```

DENGESİZ/INSTABİL İLİŞKİLER:
MOYXRFBP & AXWXFQO
MOYXRFBP & SUBJWZOO
MOYXRFBP & ESDIHDJH
MOYXRFBP & NRKJALJA
MOYXRFBP & EDKCLCLLA
MOYXRFBP & TEUQICUF
MOYXRFBP & GRGUWBGJ
MOYXRFBP & IKRPMBDT
MOYXRFBP & VAGLFUOB
MOYXRFBP & KIVVGDHT
MOYXRFBP & HYBIFMLK
MOYXRFBP & LJHUDUNG
MOYXRFBP & RRTPIOMWW
MOYXRFBP & ZCCPYSPB
AXWXFQO & BILWCDD
AXWXFQO & SAFUFGST
AXWXFQO & KEVELYFM
AXWXFQO & AWFABBTH

Şekil 8.5.2. Yapısal Dengesizlik Problemi, 1. Problem QPU/CPU Hibriti Çözümleri

Şekil 8.5.2'de birinci probleme QPU/CPU hibrit çözümleyici tarafından üretilen sonuçları gözlemleyebiliriz. Sonuçlar daha detaylı olarak Ek1'de belirtilmiştir.

Birinci problem özelinde sonuçlar incelemişinde belirli oranda örtüşmezlikler olduğu fark edilmektedir. Bu örtüşmezliklerin sebepleri ileri araştırmalar ile anlaşılabılır.

```
In [594]: for edge in si2.edges:
    si2.edges[edge]['frustrated'] = edge in qpu_imbalance_b
for node in si2.nodes:
    si2.nodes[node]['color'] = qpu_bicoloring_b[node]
|
print("2. PROBLEM (QPU):")
print('GRUP A: \n\t'.join(list(person for (person, color) in qpu_bicoloring_b.items() if (color == 0))))
print('GRUP B: \n\t'.join(list(person for (person, color) in qpu_bicoloring_b.items() if (color == 1))))
print('DENGESİZ/INSTABİL İLİŞKİLER: \n\t' + '\n\t'.join(list(x + " " + y for (x, y) in qpu_imbalance_b.keys())))
2. PROBLEM (QPU):
GRUP A:
AFIBBBHSO
AHVFUTHY
ANJQVHLF
DAZARSPK
GSHGKROF
NGGEPOJT
NUYYBHZG
VHNVBXWB
YBXSNOWQ
GRUP B:
ABGOZNAU
BLHGXFDL
HMZJINWY
KBCJPFFR
KJSNDQAN
PVPTEFBD
RISSRPHZ
XGDLWZFC
XIZKOWLP
ZAWUTIVQ
ZRUNNMYDJ
```

Şekil 8.5.3. Yapısal Dengesizlik Problemi, 2. Problem QPU Çözümleri

Şekil 8.5.3'de ikinci probleme QPU çözümleyici tarafından üretilen çözümleri gözlemlemekteyiz. Sonuçlar daha detaylı olarak Ek1'de belirtilmiştir.

```
In [597]: for edge in si2.edges:
    si2.edges[edge]['frustrated'] = edge in hybrid_imbalance_b
for node in si2.nodes:
    si2.nodes[node]['color'] = hybrid_bicoloring_b[node]

print("2. PROBLEM (QPU/CPU):")
print('GRUP A: \n\t'.join(list(person for (person, color) in hybrid_bicoloring_b.items() if (color == 0))))
print('GRUP B: \n\t'.join(list(person for (person, color) in hybrid_bicoloring_b.items() if (color == 1))))
print('DENGESİZ/INSTABİL İLİŞKİLER: \n\t'.join(list(x + " & " + y for (x, y) in hybrid_imbalance_b.keys())))

2. PROBLEM (QPU/CPU):
GRUP A:
ABGOZNAU
AHVFUTHY
ANJQVHLF
BLHGKPHD
HMZJINWY
KJSNDQAN
NGGEPOJT
PFVTEFBDO
R2SSRPHZ
XIZKOWLP
ZAUUTIVQ

GRUP B:
AFIBBBHSO
DAMARSFK
GSHGKRQF
KBCJPFFR
NUVYBHZG
VRNVBXWB
XGDLWZFC
YBKXSNOWQ
ZRUNMYDJ

DENGESİZ/INSTABİL İLİŞKİLER:
XGDLWZFC & AHVFUTHY
XGDLWZFC & ABGOZNAU
XGDLWZFC & ZAUUTIVQ
```

Şekil 8.5.4. Yapisal Dengesizlik Problemi, 2. Problem QPU/CPU Hibrit Çözümleri

Şekil 8.5.4’de ikinci probleme QPU/CPU hibrit çözümleyici tarafından getirilen çözümleri gözlemeğekteyiz. Sonuçlar daha detaylı olarak Ekl’de belirtilmiştir.

İkinci problem özelinde düşünüldüğünde, QPU/CPU hibrit ve QPU çözümeyicinin ulaştığı sonuçlar arasında örtüşmezlikler bulunmaktadır. Bu örtüşmezliklerin sebebi ileri araştırmalar ile daha net anlaşılabılır.

```
In [598]: for edge in si3.edges:
    si3.edges[edge]['frustrated'] = edge in qpu_imbalance_c
for node in si3.nodes:
    si3.nodes[node]['color'] = qpu_bicoloring_c[node]

print("3. PROBLEM (QPU):")
print('GRUP A: \n\t'.join(list(person for (person, color) in qpu_bicoloring_c.items() if (color == 0))))
print('GRUP B: \n\t'.join(list(person for (person, color) in qpu_bicoloring_c.items() if (color == 1))))
print('DENGESİZ/INSTABİL İLİŞKİLER: \n\t'.join(list(x + " & " + y for (x, y) in qpu_imbalance_c.keys())))

YUQUDZFH
DENGESİZ/INSTABİL İLİŞKİLER:
LDHLLNWQ & MZTDNRSY
LDHLLNWQ & KVCGGBPK
LDHLLNWQ & TOEMAHNU
LDHLLNWQ & QZBUYHLLX
LDHLLNWQ & SSRVFBMF
LDHLLNWQ & CF5ZAMC
LDHLLNWQ & IBJGPPLY
LDHLLNWQ & ZAJLVOILL
LDHLLNWQ & NIZLNAGM
LDHLLNWQ & PRJFPRNS
LDHLLNWQ & VOYINDVA
LDHLLNWQ & FIAHDDJC
LDHLLNWQ & VCVBBZHO
LDHLLNWQ & CZHSWOTZ
LDHLLNWQ & JRAWRBZA
LDHLLNWQ & ZUURMSLS
IAMVGBLQ & MZTDNRSY
```

Şekil 8.5.5. Yapisal Dengesizlik Problemi, 3. Problem QPU Çözümleri

Şekil 8.5.5'de üçüncü probleme QPU çözümleyici ile getirilen çözümleri inceleyebiliriz. Sonuçlar daha detaylı olarak Ek1'de belirtilmiştir.

```
In [599]: for edge in si3.edges:
    si3.edges[edge]['frustrated'] = edge in hybrid_imbalance_c
for node in si3.nodes:
    si3.nodes[node]['color'] = hybrid_bicoloring_c[node]

print("3. PROBLEM (QPU/CPU):")
print('GRUP A: \n\t'.join(list(person for (person, color) in hybrid_bicoloring_c.items() if (color == 0))))
print('GRUP B: \n\t'.join(list(person for (person, color) in hybrid_bicoloring_c.items() if (color == 1))))
print(DENGESIZ/INSTABİL İLİŞKİLER: \n\t'.join(list(x + " & " + y for (x, y) in hybrid_imbalance_c.keys())))

VÖLMLVKA
ZAJLVLOLL
DENGESIZ/INSTABİL İLİŞKİLER:
LDHLLNMQ & M2TDRNSY
LDHLLNMQ & KVCGGBTK
LDHLLNMQ & CFSZAMZC
LDHLLNMQ & CPTVVBFT
LDHLLNMQ & IBJGPPLY
LDHLLNMQ & ZAJLVLOLL
LDHLLNMQ & N1ZLNAGM
LDHLLNMQ & PRJFPNRS
LDHLLNMQ & VOYIBDVA
LDHLLNMQ & CYCMFWLY
LDHLLNMQ & WNBWBLXD
IAMVGBLQ & M2TDRNSY
IAMVGBLQ & KVCGGBTK
IAMVGBLQ & TOEMAWHU
IAMVGBLQ & QZBUYHLX
IAMVGBLQ & RTTYCLPN
IAMVGBLQ & EDCMWUFO
```

Şekil 8.5.6 Yapısal Dengesizlik Problemi, 3. Problemin QPU/CPU Hibrit Çözümleri

Şekil 8.5.6'da üçüncü probleme QPU/CPU hibrit çözümleyici ile getirilen sonuçları inceleyebiliriz. Sonuçlar daha detaylı olarak Ek1'de belirtilmiştir.

Üçüncü problem özelinde düşünüldüğünde, QPU/CPU hibrit ve QPU çözümleyicinin eriştiği sonuçlar tam anlamıyla birbiri ile örtüşmemektedir. Bu örtüşmezliğin sebepleri ileri araştırmalarla anlaşılabılır.

Devam eden başlıkta çalışmanın deney süreci dahilinde ulaşılan sonuçlar tartışılabacak ve elde edilen veriler doğrultusunda alanda atılabilcek bir sonraki adımlardan bahsedilecektir.

9. TARTIŞMA

Çalışmamızda ulaştığımız sonuçların D-Wave'in QPU ve QPU/CPU hibrit çözümleyicilerinin sektörel kullanımına ilişkin fikir verebilmesi amaçlanmıştır. Bu sayede çeşitli yüneylem problemlerinin çözümlenmesi için klasik CPU'lara ek olarak kuantum bilgisayarlardan da fayda sağlanabileceği yeni bir çağ'a giriş yapılabilir.

Deneyselimizde ulaştığımız sonuçlar kuantum bilgisayarlarının hala geliştirilmeye ve hatasızlaştırılmaya ihtiyaç duyduğunu ortaya koymaktadır. Çözümlemeler sonucundaki hata oranlarının daha az bir düzeye indirilmesi, kuantum bilgisayarların hızının önemini daha da artıracak ve çeşitli ticari alanlarda yeni kullanım alanları yaratacaktır. Bu cihazların ticari kullanımının mümkünüğünün test edilmesi için, özellikle klasik CPU çözümleyicilerin sonuçları ile kıyaslandığında görülen örtüşmezliklerin sebebinin iyice araştırılması gerekmektedir. Bu sayede bu örtüşmezlikler ortadan kaldırılarak daha tutarlı ve hızlı sonuçlara ulaşılması kolaylaşır.

Benzer alanda çalışma yapmak isteyen akademik ve ticari çevrelere öneremek en önemli aşama, CPU ve QPU arasındaki çözümleme farklılıklarını araştırmak ve analiz etmektir. Buna ek olarak gerçek dünyada rastlanan örnek problemlerin, çalışmamızdakine benzer bir şekilde ifade edilerek çeşitli farklı yöntemler ile analiz edilmesi ve performans ölçümlemelerinin yapılması da kuantum bilgisayarlarının fizibilitesinin test edilmesi konusunda önemli bir basamak olacaktır.

10. SONUÇ

Çalışmamız dahilinde gerçekleştirdiğimiz deneylerde, QPU ve QPU/CPU hibritinin birçok noktada CPU'ya kıyasla daha hızlı olduğunu gözlemledik. QPU ve QPU/CPU hibritinin, CPU'ya kıyasla daha avantajlı olduğu bir problem boyutu bulunmaktadır. Bu noktaya kadar, CPU daha avantajlı iken, bu noktanın ardından CPU'nun işlem süresi üssel bir artışa girmekte ve bir noktadan sonra fizibilitesi olmayan sürelerde olaşmaktadır. Bu tür noktalarda QPU ve QPU/CPU servisleri oldukça hızlı sonuçlar üretmekteydi.

Elbette CPU'lar yerine QPU'ların yöneylem araştırmalarında yaygınlaşmasını sağlayabilmek için hız kadar tutarlılık ve doğruluk da önemli bir faktördür. Deneylerimiz süresinde QPU ve CPU'nun ürettiği çözümlerin her zaman tam anlamıyla birbiri ile örtüşmediğini gözlemediğimiz. Bu örtüşmezliklerin sebebi gürültü, tekil ölçümleme veya henüz QPU donanımının yeteri kadar gelmemesi olabilir. Çoklu ölçümlemelerin, olasılıksal çalışma prensibine sahip kuantum bilgisayarların çözümlerinin doğruluğunu artırması mümkün olabilir. Mevcut teknolojik düzeyde QPU ve QPU/CPU hibriti tarafından üretilen çözümlemelerinin birbiri ile örtüşmemesinin sebepleri daha etkin bir biçimde ileri çalışmalarca araştırılmalıdır.

QPU'ya ek olarak QPU/CPU hibrit çözümleyiciler günümüzdeki orta vadeli zaman dilimi içerisinde kuantum bilgisayarların sektörel kullanımında önemlidir. Bunun en temel sebebi, mevcut kuantum bilgisayar teknolojisinin henüz emekleme düzeyinde olması, ve birçok noktada halen CPU'ların işlem gücünün yüksek fayda sağlayabilmesidir. QPU ve CPU'ların güçlü taraflarının uygun alt yapılar ile birleştirilmesi, kuantum bilgisayar teknolojilerinin gelişme sağladığı süre boyunca sektördeki birçok alanda avantaj sağlayabilir.

Kuantum bilgisayarların performanslarını direkt olarak etkileyen bir diğer faktör olan Qubit sayısının da zaman içerisinde artması, gürültü düzeyinin de azaltılabilıldığı varsayıldığında kuantum bilgisayarların avantajını gittikçe artıracak ve birçok sektörde alanda kullanımlarını hızlandıracaktır.

D-Wave'in hizmete sunduğu analog kuantum bilgisayarlara ek olarak, dijital kuantum bilgisayarları da çok büyük gelecek vaat etmektedir. Önümüzdeki 30-40 yılda dijital kuantum bilgisayarları daha etkin çalışabilecekleri noktalara ulaşacak ve bu alanda yatırımda bulunan ülkelere ve şirketlere çok büyük geri dönüşler sağlayacaktır.

Çalışmamızda rastladığımız bir diğer kısıtlayıcı faktör, kuantum bilgisayarlara bulut üzerinden erişim sağlamamızın sebep olduğu ağ kaynaklı gecikmelerdir. Bu gecikmeler çoğu zaman önceden tahminlenemez olmakta ancak problemlere ilişkin ürettiğimiz çözümleme sürecinin çok büyük bir kısmını oluşturmaktadır. Ağ kaynaklı gecikmeler hesaba katılmadığında kuantum bilgisayarların mevcut problemleri çözümlemesi CPU'ya kıyasla çok daha hızlı olabilmektedir. Biz çalışmamız dahilinde toplam çözüm süresini baz aldığımızdan, ağ kaynaklı gecikmeleri de süreye dahil etmeyi tercih ettik.

Çalışmamız süresince şaşırtıcı olan sonuçlardan biri, belirli problemlerde QPU/CPU hibritinin QPU'ya kıyasla daha yavaş çözümleme gerçekleştirmesi idi. Bunun temel sebebi, ürettiğimiz problemin QPU/CPU altyapısı ile uyumsuzluğu olabileceği gibi, daha ileri araştırmalar ile anlaşılabilen başka sebepler de olabilir.

Son olarak kuantum bilgisayarların ilerleyen teknoloji ile birlikte mevcut CPU'lar ile ya çözülmesi çok maliyetli ya da hiç mümkün olmayan problemlere çözüm getirmesine yakın gelecekte şahit olacağız. Çalışmamız dahilinde bunu küçük örneklerle ifade etmiş olduk ve konu hakkında daha detaylı araştırmada bulunmak isteyen akademik ve ticari çevrelere ilham kaynağı olmayı hedefledik.

KAYNAKLAR DİZİNİ

- Moore, Gordon E.** (1965-04-19). "Cramming more components onto integrated circuits" (PDF). intel.com. Electronics Magazine. Retrieved April 1, 2020.
- Moore, Gordon E.** (1975). "Progress in digital integrated electronics" (http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1478174). IEEE. . Retrieved 2011-11-27.
- Ieeexplore.ieee.org.** 2021. “Moore's Law: Past, Present And Future” - IEEE Journals & Magazine. [online] Available at: <<https://ieeexplore.ieee.org/abstract/document/591665>> [Accessed 7 January 2021].
- Moore, Gordon E.** 2021. “Lithography And The Future Of Moore's Law”. Accessed 7 January 2021.
- Schaller, Bob** (September 26, 1996). "The Origin, Nature, and Implications of "MOORE'S LAW"". Microsoft. Retrieved September 10, 2014.
- John L. Hennessy; David A. Patterson** (June 4, 2018). "A New Golden Age for Computer Architecture: Domain-Specific Hardware/Software Co-Design, Enhanced Security, Open Instruction Sets, and Agile Chip Development". International Symposium on Computer Architecture - ISCA 2018. In the later 1990s and 2000s, architectural innovation decreased, so performance came primarily from higher clock rates and larger caches. The ending of Dennard Scaling and Moore's Law also slowed this path; single core performance improved only 3% last year!
- Sciedirect.com.** 2021. “Quantum Size Effect - An Overview” | Sciedirect Topics. [online] Available at: <<https://www.sciencedirect.com/topics/chemistry/quantum-size-effect>> [Accessed 7 January 2021].
- Saracco, R.**, 2021. “Tiniest Transistor – Yet: 2.5 Nm - IEEE Future Directions.” [online] IEEE Future Directions. Available at: <<https://cmte.ieee.org/futuredirections/2019/03/13/tiniest-transistor-yet-2-5-nm/>> [Accessed 7 January 2021].
- Basdevant, J.-L.; Rich, J.; Spiro, M.** (2005). “Fundamentals in Nuclear Physics. Springer.” p. 13, fig 1.1. ISBN 978-0-387-01672-6.
- Lee, Changgu** (2008). "Measurement of the Elastic Properties and Intrinsic Strength of Monolayer Graphene". Science. 321 (385): 385–388. Bibcode:2008Sci...321..385L. doi:10.1126/science.1157996. PMID 18635798. S2CID 206512830.
- Benioff, Paul** (1980). "The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines". Journal of Statistical Physics. 22 (5): 563–591. Bibcode:1980JSP....22..563B. doi:10.1007/bf01011339. S2CID 122949592.

KAYNAKLAR DİZİNİ (devam)

- Feynman, Richard** (June 1982). "Simulating Physics with Computers"(PDF). International Journal of Theoretical Physics. 21 (6/7): 467–488. Bibcode:1982IJTP...21..467F. doi:10.1007/BF02650179. S2CID 124545445. Archived from the original (PDF) on 8 January 2019. Retrieved 28 February 2019.
- Mermin, David** (March 28, 2006). "Breaking RSA Encryption with a Quantum Computer: Shor's Factoring Algorithm" (PDF). Physics 481-681 Lecture Notes. Cornell University. Archived from the original (PDF) on 2012-11-15.
- Schrödinger, Erwin** (November 1935). "Die gegenwärtige Situation in der Quantenmechanik (The present situation in quantum mechanics)". Naturwissenschaften. 23 (48): 807–812. Bibcode:1935NW.....23..807S. doi:10.1007/BF01491891. S2CID 206795705
- García-Martín, D. and Sierra, G.**, 2021. “Five Experimental Tests On The 5-Qubit IBM Quantum Computer.” Accessed 7 January 2021.
- Jones, J., Mosca, M. and Hansen, R.**, 2021. “Implementation Of A Quantum Search Algorithm On A Quantum Computer.” Accessed 7 January 2021.
- Schuld, M., Bocharov, A., Svore, K. and Wiebe, N.**, 2021. “Circuit-Centric Quantum Classifiers.” Accessed 7 January 2021.
- Yang, C., Leon, R., Hwang, J., Saraiva, A., Tanttu, T., Huang, W., Camirand Lemyre, J., Chan, K., Tan, K., Hudson, F., Itoh, K., Morello, A., Pioro-Ladrière, M., Laucht, A. and Dzurak, A.**, 2021. “Operation Of A Silicon Quantum Processor Unit Cell Above One Kelvin.” Accessed 7 January 2021.
- Cartwright, D.; Harary, Frank** (1956). "Structural balance: a generalization of Heider's theory" (PDF). Psychological Review. 63 (5): 277–293. doi:10.1037/h0046049.
- Cao, Jiayu & Fan, Ying & Di, Zengru**. (2018). “Frustration of signed networks: How does it affect the thermodynamic properties of a system?.”
- Puccia, Charles J. and Levins, Richard** (1986). Qualitative Modeling of Complex Systems: An Introduction to Loop Analysis and Time Averaging. Harvard University Press, Cambridge, MA.
- Kunegis, Jérôme & Schmidt, Stephan & Lommatzsch, Andreas & Lerner, Jürgen & De Luca, Ernesto & Albayrak, Sahin.** (2010). Spectral Analysis of Signed Graphs for Clustering, Prediction and Visualization. Proc SDM. 559-. 10.1137/1.9781611972801.49.
- Journals.uchicago.edu.** 2021. “The Structural Balance Theory Of Sentiment Networks: Elaboration And Test” | American Journal Of Sociology: Vol 123, No 2. [online] Available at: <<https://www.journals.uchicago.edu/doi/full/10.1086/692757>> [Accessed 7 January 2021].

KAYNAKLAR DİZİNİ (devam)

David Deutsch & Richard Jozsa (1992). "Rapid solutions of problems by quantum computation". Proceedings of the Royal Society of London A. 439 (1907): 553–558. Bibcode: 1992RSPSA.439..553D. CiteSeerX 10.1.1.655.5997. doi:10.1098/rspa.1992.0167.

Ethan Bernstein and Umesh Vazirani (1997). "Quantum Complexity Theory". SIAM Journal on Computing. 26 (5): 1411–1473. doi:10.1137/S0097539796300921

Daniel R. Simon (1997) "On the Power of Quantum Computation" SIAM Journal on Computing, 26(5), 1474–1483, doi:10.1137/S0097539796298637

Arute, F., Arya, K., Babbush, R. et al. "Quantum supremacy using a programmable superconducting processor." Nature 574, 505–510 (2019). <https://doi.org/10.1038/s41586-019-1666-5>

On "Quantum Supremacy"". IBM Research Blog. 2019-10-22. Retrieved 2019-10-24.

D. Willsch, M. Willsch, H. De Raedt, K. Michielsen, "Support vector machines on the D-Wave quantum annealer", Computer Physics Communications, Volume 248, 2020, 107006, ISSN 0010-4655, <https://doi.org/10.1016/j.cpc.2019.107006>. (<http://www.sciencedirect.com/science/article/pii/S001046551930342X>)

D. Willsch, M. Willsch, H. De Raedt, K. Michielsen, Support vector machines on the D-Wave quantum annealer, Computer Physics Communications, Volume 248, 2020, 107006, ISSN 0010-4655, <https://doi.org/10.1016/j.cpc.2019.107006>. (<http://www.sciencedirect.com/science/article/pii/S001046551930342X>)

ÖZGEÇMİŞ

Adı-Soyadı : Ege Doğan Dursun

Doğum Tarihi/Yeri : 12.04.1997, İzmir

Eğitim

İlköğretim : Karşıyaka İlköğretim Okulu

Ortaöğretim : Karşıyaka İlköğretim Okulu

Lise : Bornova Anadolu Lisesi

Üniversite : Ege Üniversitesi, Bilgisayar Mühendisliği Bölümü

Sürekli Adres : Ataşehir Mahallesi. Nazım Hikmet Ran Bulvarı No:18

Daire:1 Çiğli İzmir TÜRKİYE 35620

Telefon : +90 (507) 055 8665

E-posta : edogandursun@gmail.com

Vatandaşlık : Türkiye

EKLER

Ek 1 Jupyter Notebook Çalışma Kod Dökümanları

D-Wave Kuantum Tavlama Servisleriyle Yapısal Dengesizlik Problemlerine İlişkin Çözümlerin Fizibilitesini İnceleme

EGE ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

2020 - 2021 ÖĞRETİM YILI

(LİSANS TEZİ)

DURSUN, EGE DOĞAN

ÇORBACIOĞLU, CEM

TEZ DANIŞMANI: DOÇ. DR. MURAT OSMAN ÜNALIR

NetworkX Çizge Örnekleri

Bu bölümde, NetworkX kütüphanesi kullanılarak yaratılabilen çeşitli çizgelere örnekler verilmiştir. Çalışmanın ilerleyen bölümlerinde, yine NetworkX ve D-Wave NetworkX kütüphanesi dahilinde yaratılan çizgeler kullanılarak çeşitli Yöneylem problemleri simülle edilmeye ve çözümlenmeye çalışılmıştır.

```
In [1]: import networkx as nx
import matplotlib.pyplot as plt
%matplotlib inline
```

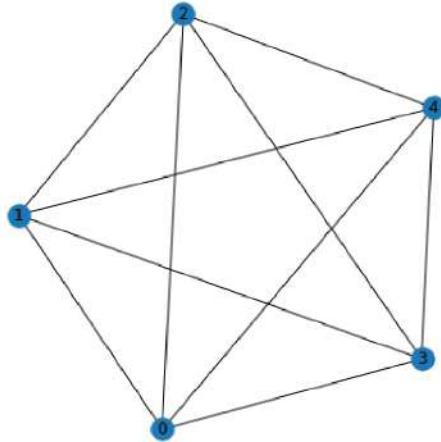
Complete Graphs

04/02/2021

TEZ_DOC

```
In [2]: cg1 = nx.complete_graph(5)
plt.figure(figsize=(5,5))
nx.draw(cg1, with_labels=True, font_weights='bold')

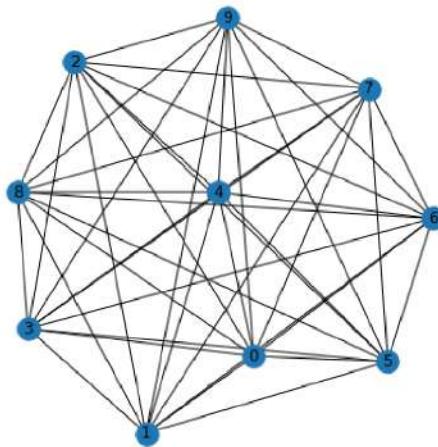
/Users/egedursun/opt/anaconda3/lib/python3.7/site-packages/networkx/drawing/nx_pylab.py:579: MatplotlibDeprecationWarning:
The iterable function was deprecated in Matplotlib 3.1 and will be removed in 3.3. Use np.iterable instead.
    if not cb.iterable(width):
```



04/02/2021

TEZ_DOC

```
In [3]: cg2 = nx.complete_graph(10)
plt.figure(figsize=(5,5))
nx.draw(cg2, with_labels=True, font_weights='bold')
```

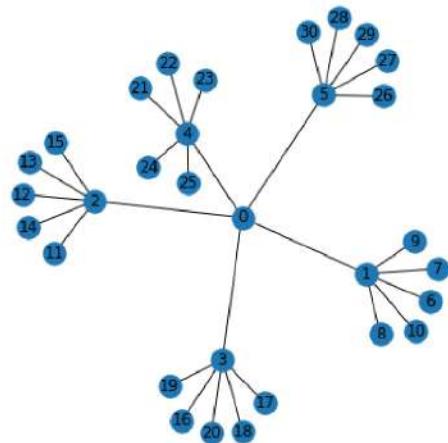


Balanced Tree

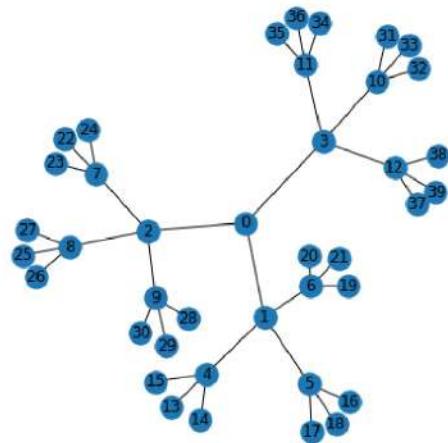
04/02/2021

TEZ_DOC

```
In [4]: btl = nx.balanced_tree(5, 2)
plt.figure(figsize=(5,5))
nx.draw(btl, with_labels=True, font_weights='bold')
```

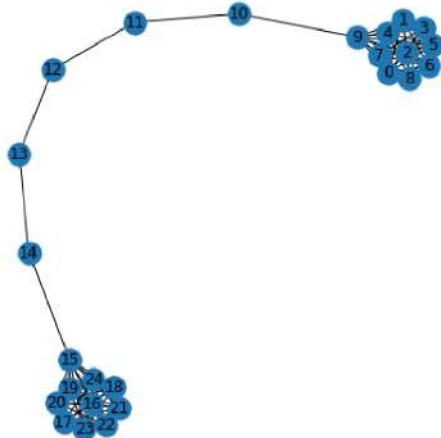


```
In [5]: bt2 = nx.balanced_tree(3, 3)
plt.figure(figsize=(5,5))
nx.draw(bt2, with_labels=True, font_weights='bold')
```



Barbell Graph

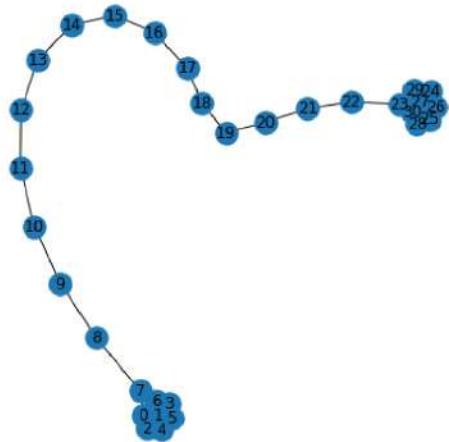
```
In [6]: bgl = nx.barbell_graph(10, 5)
plt.figure(figsize=(5,5))
nx.draw(bgl, with_labels=True, font_weights='bold')
```



04/02/2021

TEZ_DOC

```
In [7]: bg2 = nx.barbell_graph(8, 15)
plt.figure(figsize=(5,5))
nx.draw(bg2, with_labels=True, font_weights='bold')
```

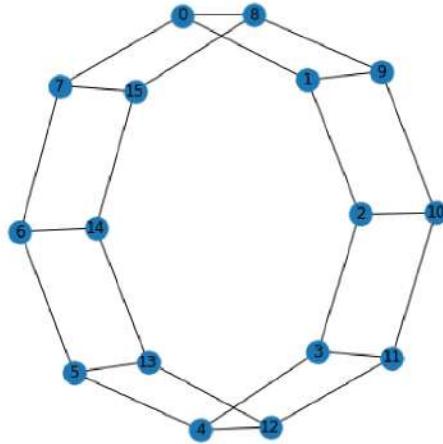


Circular Ladder Graph

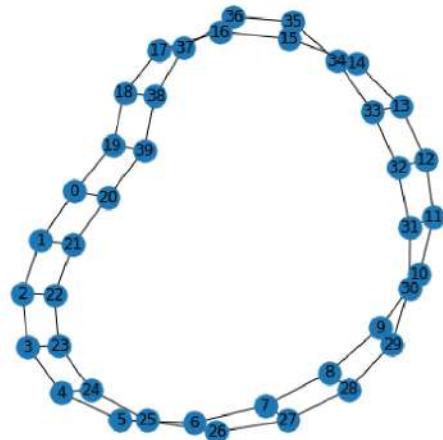
04/02/2021

TEZ_DOC

```
In [8]: cl1 = nx.circular_ladder_graph(8)
plt.figure(figsize=(5,5))
nx.draw(cl1, with_labels=True, font_weights='bold')
```

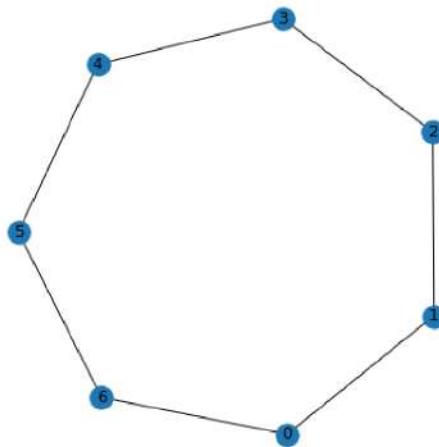


```
In [9]: cl2 = nx.circular_ladder_graph(20)
plt.figure(figsize=(5,5))
nx.draw(cl2, with_labels=True, font_weights='bold')
```



Cycle Graph

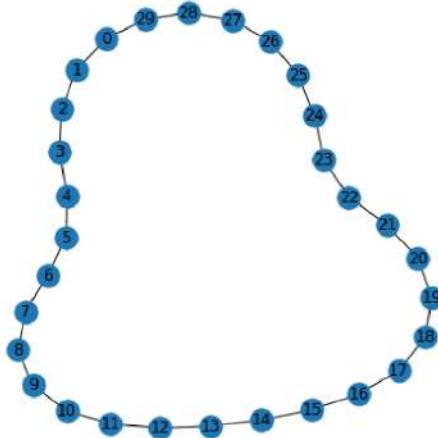
```
In [10]: cyg1 = nx.cycle_graph(7)
plt.figure(figsize=(5,5))
nx.draw(cyg1, with_labels=True, font_weights='bold')
```



04/02/2021

TEZ_DOC

```
In [11]: cyg2 = nx.cycle_graph(30)
plt.figure(figsize=(5,5))
nx.draw(cyg2, with_labels=True, font_weights='bold')
```

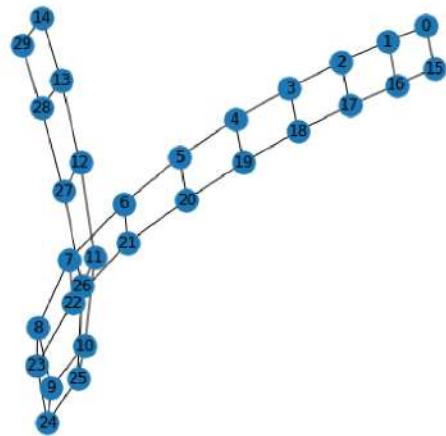


Ladder Graph

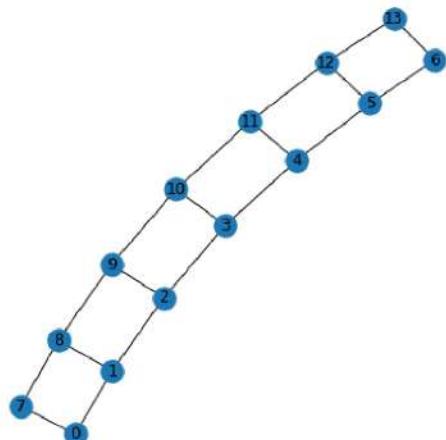
04/02/2021

TEZ_DOC

```
In [12]: lag1 = nx.ladder_graph(15)
plt.figure(figsize=(5,5))
nx.draw(lag1, with_labels=True, font_weights='bold')
```

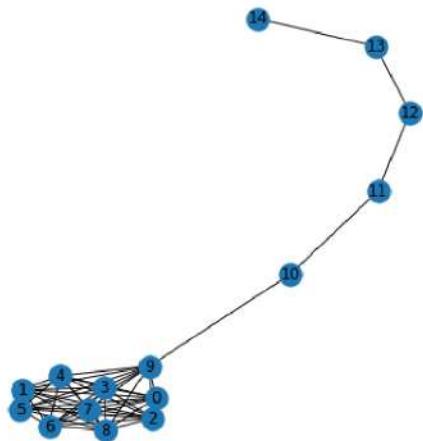


```
In [13]: lag2 = nx.ladder_graph(7)
plt.figure(figsize=(5,5))
nx.draw(lag2, with_labels=True, font_weights='bold')
```



Lollipop Graph

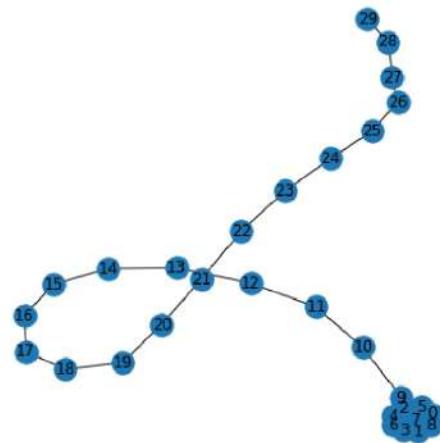
```
In [14]: log1 = nx.lollipop_graph(10, 5)
plt.figure(figsize=(5,5))
nx.draw(log1, with_labels=True, font_weights='bold')
```



04/02/2021

TEZ_DOC

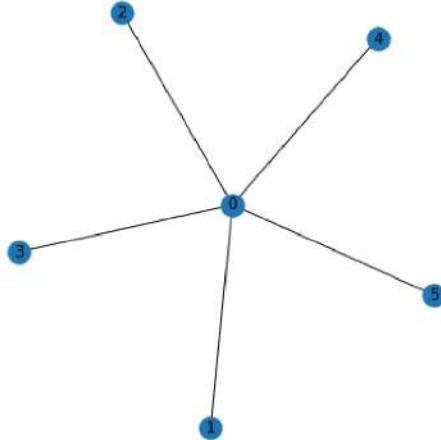
```
In [15]: log2 = nx.lollipop_graph(10, 20)
plt.figure(figsize=(5,5))
nx.draw(log2, with_labels=True, font_weights='bold')
```

**Star Graph**

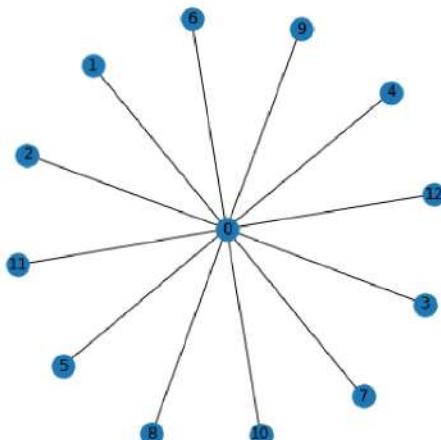
04/02/2021

TEZ_DOC

```
In [16]: stg1 = nx.star_graph(5)
plt.figure(figsize=(5,5))
nx.draw(stg1, with_labels=True, font_weights='bold')
```

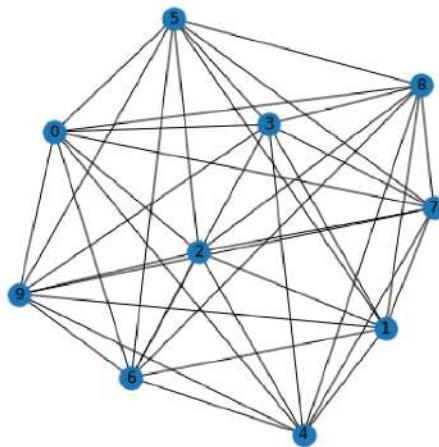


```
In [17]: stg2 = nx.star_graph(12)
plt.figure(figsize=(5,5))
nx.draw(stg2, with_labels=True, font_weights='bold')
```



Turan Graph

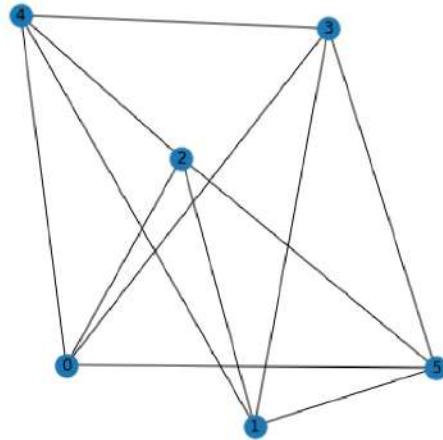
```
In [18]: tr1 = nx.turan_graph(10, 5)
plt.figure(figsize=(5,5))
nx.draw(tr1, with_labels=True, font_weights='bold')
```



04/02/2021

TEZ_DOC

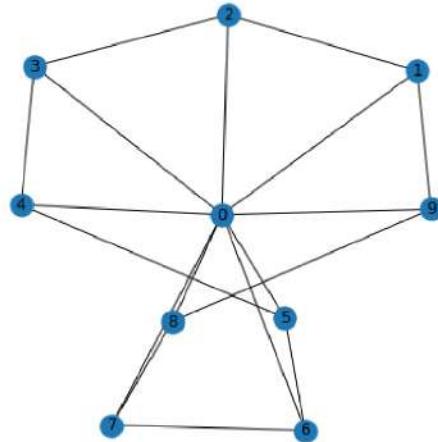
```
In [19]: tr2 = nx.turan_graph(6, 3)
plt.figure(figsize=(5,5))
nx.draw(tr2, with_labels=True, font_weights='bold')
```

**Wheel Graph**

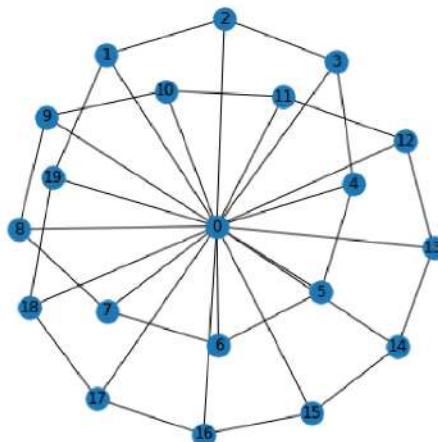
04/02/2021

TEZ_DOC

```
In [20]: whg1 = nx.wheel_graph(10)
plt.figure(figsize=(5,5))
nx.draw(whg1, with_labels=True, font_weights='bold')
```



```
In [21]: whg2 = nx.wheel_graph(20)
plt.figure(figsize=(5,5))
nx.draw(whg2, with_labels=True, font_weights='bold')
```



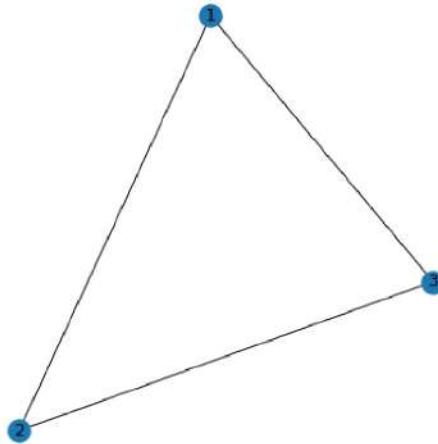
localhost:8888/nbconvert/html/Macintosh HD/Users/egedursun/Desktop/Tez/TEZ_DOC.ipynb?download=false

16/146

Özel Ağırlıklı Çizge Yaratmak

```
In [24]: spec_graph = nx.Graph()
spec_graph.add_edge(1,2, weight=7)
spec_graph.add_edge(2,3, weight=9)
spec_graph.add_edge(3,1, weight=17)

plt.figure(figsize=(5,5))
nx.draw(spec_graph, with_labels=True, font_weights='bold')
```



D-Wave QPU Çözümleyiciler ile Çeşitli Problemlerin Çözümlenmesi

Bu bölümde NetworkX kütüphanesi kullanılarak yaratılmış çizgeler ile ifade edilen problemleri, D-Wave'in bulut üzerinden sunduğu Kuantum Tavlama servisleri ile çözümlemeye çalıştık. QPU ile elde ettiğimiz sonuçları, CPU performansı ile kıyaslayarak farklarını raporlandırdık.

```
In [48]: #CPU Çözümleyicisinin import edilmesi
from dimod.reference.samplers import ExactSolver

#QPU Çözümleyicisinin import edilmesi
from dwave.system.samplers import DWaveSampler

from dwave.system.composites import EmbeddingComposite
import dwave_networkx as dnx

import time
import random
```

04/02/2021

TEZ_DOC

```
In [49]: #CPU çözümleyicisini yarat
cpu_sampler = ExactSolver()

#QPU çözümleyicisini yarat
cpu_sampler = EmbeddingComposite(DWaveSampler())
```

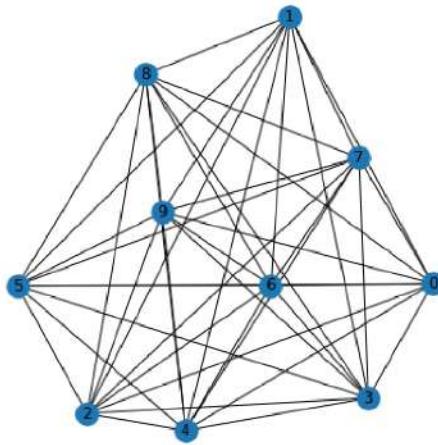
Minimum Vertex Cover Problemi

Birinci Problem

```
In [30]: vcv1 = nx.complete_graph(10)

#Bağlantıların ağırlıklarının rastgele belirlenmesi
for (u,v,w) in vcv1.edges(data=True):
    w['weight'] = random.randint(0,20)

plt.figure(figsize=(5,5))
nx.draw(vcv1, with_labels=True, font_weights='bold')
```



İkinci Problem

localhost:8888/nbconvert/html/Macintosh HD/Users/egedursun/Desktop/Tez/TEZ_DOC.ipynb?download=false

18/146

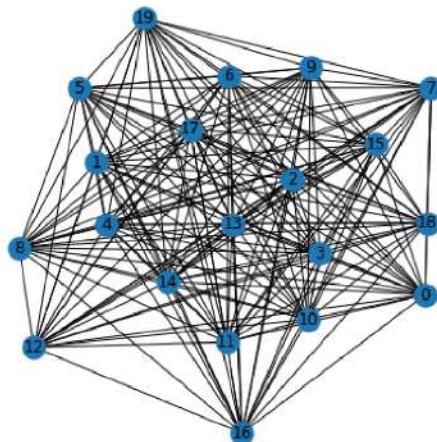
04/02/2021

TEZ_DOC

```
In [31]: vcv2 = nx.complete_graph(20)

for (u,v,w) in vcv2.edges(data=True):
    w['weight'] = random.randint(0,20)

plt.figure(figsize=(5,5))
nx.draw(vcv2, with_labels=True, font_weights='bold')
```



Üçüncü Problem

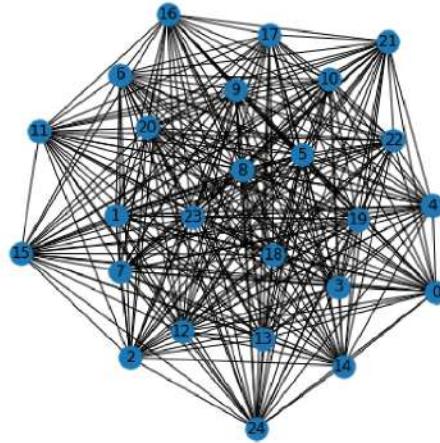
04/02/2021

TEZ_DOC

```
In [32]: vcv3 = nx.complete_graph(25)

for (u,v,w) in vcv3.edges(data=True):
    w['weight'] = random.randint(0,20)

plt.figure(figsize=(5,5))
nx.draw(vcv3, with_labels=True, font_weights='bold')
```



Birinci Problem - QPU ile Çözümleme

```
In [33]: q_start = time.time()

qpu_result_a = dnx.min_vertex_cover(vcv1, qpu_sampler)

q_end = time.time()

qpu_result_time = (q_end - q_start)

print("1. PROBLEM - QPU ÇÖZÜM SÜRESİ: ",qpu_result_time)
```

1. PROBLEM - QPU ÇÖZÜM SÜRESİ: 5.6201417446136475

Birinci Problem - CPU ile Çözümleme

04/02/2021

TEZ_DOC

```
In [34]: c_start = time.time()
cpu_result_a = dnx.min_vertex_cover(vcv1, cpu_sampler)
c_end = time.time()
cpu_result_time = (c_end - c_start)
print("1. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)
```

1. PROBLEM - CPU ÇÖZÜM SÜRESİ: 0.004764080047607422

İkinci Problem - QPU ile Çözümleme

```
In [35]: q_start = time.time()
qpu_result_b = dnx.min_vertex_cover(vcv2, qpu_sampler)
q_end = time.time()
qpu_result_time = (q_end - q_start)
print("2. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)
```

2. PROBLEM - QPU ÇÖZÜM SÜRESİ: 5.60565972328186

İkinci Problem - CPU ile Çözümleme

```
In [36]: c_start = time.time()
cpu_result_b = dnx.min_vertex_cover(vcv2, cpu_sampler)
c_end = time.time()
cpu_result_time = (c_end - c_start)
print("2. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)
```

2. PROBLEM - CPU ÇÖZÜM SÜRESİ: 3.043724775314331

Üçüncü Problem - QPU ile Çözümleme

04/02/2021

TEZ_DOC

```
In [37]: q_start = time.time()

qpu_result = dnx.min_vertex_cover(vcv3, qpu_sampler)

q_end = time.time()

qpu_result_time = (q_end - q_start)

print("3. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)
```

3. PROBLEM - QPU ÇÖZÜM SÜRESİ: 7.12906289100647

Üçüncü Problem - CPU ile Çözümleme

```
In [38]: c_start = time.time()

cpu_result = dnx.min_vertex_cover(vcv3, cpu_sampler)

c_end = time.time()

cpu_result_time = (c_end - c_start)

print("3. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)
```

3. PROBLEM - CPU ÇÖZÜM SÜRESİ: 220.4924087524414

Birinci Problem - Çözümlerin Karşılaştırılması

```
In [39]: print("1. PROBLEM - QPU SONUÇ: ", qpu_result_a)
print("1. PROBLEM - CPU SONUÇ: ", cpu_result_a)
```

1. PROBLEM - QPU SONUÇ: [1, 2, 3, 4, 5, 7, 8, 9]
 1. PROBLEM - CPU SONUÇ: [0, 1, 2, 3, 4, 5, 6, 7, 9]

İkinci Problem - Çözümlerin Karşılaştırılması

```
In [40]: print("2. PROBLEM - QPU SONUÇ: ", qpu_result_b)
print("2. PROBLEM - CPU SONUÇ: ", cpu_result_b)
```

2. PROBLEM - QPU SONUÇ: [0, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 15, 16]
 2. PROBLEM - CPU SONUÇ: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
 14, 15, 16, 17, 19]

Üçüncü Problem - Çözümlerin Karşılaştırılması

04/02/2021

TEZ_DOC

```
In [41]: print("3. PROBLEM - QPU SONUÇ: ", qpu_result)
print("3. PROBLEM - CPU SONUÇ: ", cpu_result)

3. PROBLEM - QPU SONUÇ: [2, 3, 4, 5, 6, 8, 9, 12, 14, 15, 16, 17, 19,
21, 24]
3. PROBLEM - CPU SONUÇ: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
14, 15, 16, 17, 18, 19, 20, 21, 22, 24]
```

=====

CPU GECİKMESİNİ ETKİLEYEN FAKTÖRLER:

- CPU FREKANSI
- CPU ÇEKİRDEK SAYISI
- RAM BOYUTU

QPU GECİKMESİNİ ETKİLEYEN FAKTÖRLER:

- HTTP REQUEST İLETİM GECİKMESİ
- HTTP RESPONSE İLETİM GECİKMESİ

=====

N (nod-sayısı)	CPU gecikmesi (saniye)	QPU gecikmesi (saniye)
1	0.0	2.37
2	0.49	1.92
3	0.0	1.90
4	0.0	1.90
5	0.0	1.91
6	0.1	1.91
7	0.0	1.92
8	0.0	1.95
9	0.0	1.95
10	0.2	2.00
11	0.3	2.12
12	0.64	2.06
13	0.15	2.04
14	0.26	2.08
15	0.37	2.21
16	0.83	2.19
17	1.91	2.19
18	3.37	2.40
19	7.26	2.53
20	13.8	2.42
21	28.6	2.86
22	56.1	2.76
23	120.4	2.32

04/02/2021

TEZ_DOC

N (nod-sayisi)	CPU gecikmesi (saniye)	QPU gecikmesi (saniye)
24	253.5	2.12
25	508.2	2.82
26	1103.7	2.87
27	2343.2	1.92

04/02/2021

TEZ_DOC

```
In [71]: dot_amount = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27]

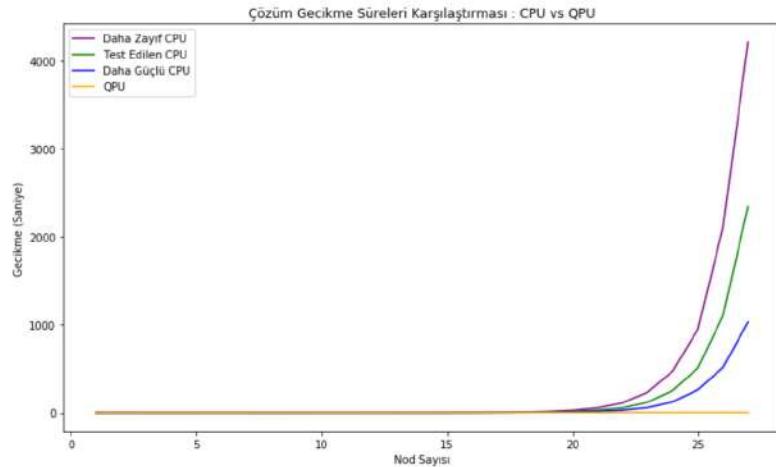
weaker_cpu_latency = [0.0, 0.0, 0.0, 0.0, 0.0, 0.1, 0.01, 0.00, 0.02, 0.01, 0.05, 0.08, 0.14, 0.32, 0.72, 1.47, 3.12, 6.84, 13.48, 27.11, 55.84, 112.27, 234.89, 474.39, 949.23, 2104.35, 4211.43]

cpu_latency = [0.0, 0.49, 0.0, 0.0, 0.0, 0.1, 0.0, 0.0, 0.0, 0.2, 0.3, 0.64, 0.15, 0.26, 0.37, 0.83, 1.91, 3.37, 7.26, 13.8, 28.6, 56.1, 120.4, 253.5, 508.2, 1103.7, 2343.2]

stronger_cpu_latency = [0.0, 0.0, 0.0, 0.01, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.01, 0.2, 0.11, 0.3, 0.63, 0.88, 1.82, 3.39, 8.21, 14.3, 29.54, 58.5, 124.5, 261.7, 514.7, 1031.14]

gpu_latency = [2.37, 1.92, 1.90, 1.90, 1.91, 1.91, 1.92, 1.95, 1.95, 2.00, 2.12, 2.06, 2.04, 2.08, 2.21, 2.19, 2.19, 2.40, 2.53, 2.42, 2.86, 2.76, 2.32, 2.12, 2.82, 2.87, 1.92]

plt.figure(figsize=(12,7))
plt.plot(dot_amount, weaker_cpu_latency, color="purple")
plt.plot(dot_amount, cpu_latency, color='g')
plt.plot(dot_amount, stronger_cpu_latency, color="blue")
plt.plot(dot_amount, gpu_latency, color='orange')
plt.xlabel('Nod Sayısı')
plt.ylabel('Gecikme (Saniye)')
plt.title('Çözüm Gecikme Süreleri Karşılaştırması : CPU vs QPU')
plt.legend(['Daha Zayıf CPU','Test Edilen CPU', 'Daha Güçlü CPU', 'QPU'])
plt.show()
```



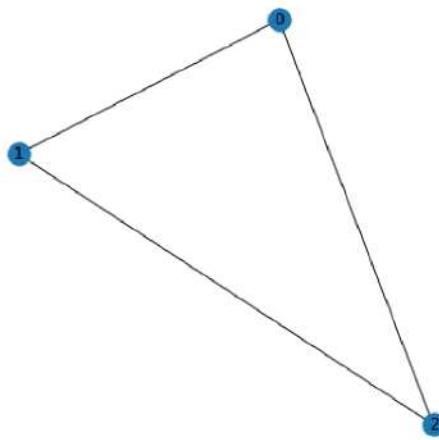
Map Coloring Problemı

Birinci Problem

```
In [57]: mcv1 = nx.complete_graph(3)

for (u,v,w) in mcv1.edges(data=True):
    w['weight'] = random.randint(0,20)

plt.figure(figsize=(5,5))
nx.draw(mcv1, with_labels=True, font_weights='bold')
```



İkinci Problem

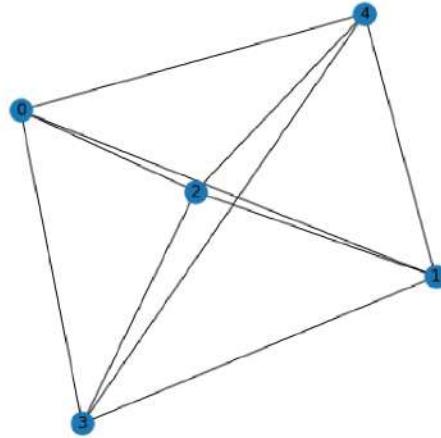
04/02/2021

TEZ_DOC

```
In [58]: mcv2 = nx.complete_graph(5)

for (u,v,w) in mcv2.edges(data=True):
    w['weight'] = random.randint(0,20)

plt.figure(figsize=(5,5))
nx.draw(mcv2, with_labels=True, font_weights='bold')
```



Üçüncü Problem

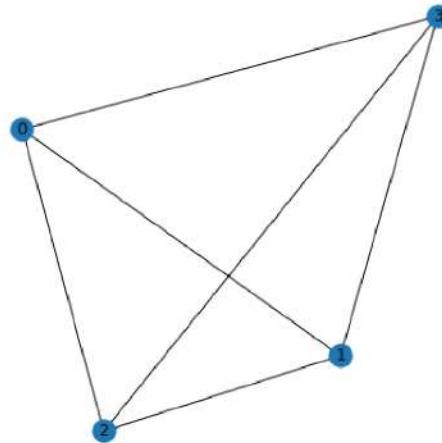
04/02/2021

TEZ_DOC

```
In [76]: mcv3 = nx.complete_graph(4)

for (u,v,w) in mcv3.edges(data=True):
    w['weight'] = random.randint(0,20)

plt.figure(figsize=(5,5))
nx.draw(mcv3, with_labels=True, font_weights='bold')
```



Birinci Problem - QPU ile Çözümleme

```
In [64]: c_start = time.time()

qpu_result_a = dnx.min_vertex_color(mcv1, qpu_sampler)

c_end = time.time()

qpu_result_time = (c_end - c_start)

print("1. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)
```

1. PROBLEM - QPU ÇÖZÜM SÜRESİ: 5.169461011886597

Birinci Problem - CPU ile Çözümleme

localhost:8888/nbconvert/html/Macintosh HD/Users/egedursun/Desktop/Tez/TEZ_DOC.ipynb?download=false

29/146

04/02/2021

TEZ_DOC

```
In [65]: c_start = time.time()
cpu_result_a = dnx.min_vertex_color(mcv1, cpu_sampler)
c_end = time.time()
cpu_result_time = (c_end - c_start)
print("1. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)
```

1. PROBLEM - CPU ÇÖZÜM SÜRESİ: 0.0037550926208496094

İkinci Problem - QPU ile Çözümleme

```
In [601]: c_start = time.time()
qpu_result_b = dnx.min_vertex_color(mcv2, qpu_sampler)
c_end = time.time()
qpu_result_time = (c_end - c_start)
print("2. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)
```

2. PROBLEM - QPU ÇÖZÜM SÜRESİ: 5.625643014907837

İkinci Problem - CPU ile Çözümleme

```
In [602]: c_start = time.time()
cpu_result_b = dnx.min_vertex_color(mcv2, cpu_sampler)
c_end = time.time()
cpu_result_time = (c_end - c_start)
print("2. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)
```

2. PROBLEM - CPU ÇÖZÜM SÜRESİ: 110.11295008659363

Üçüncü Problem - QPU ile Çözümleme

04/02/2021

TEZ_DOC

```
In [600]: c_start = time.time()
qpu_result = dnx.min_vertex_color(mcv3, qpu_sampler)
c_end = time.time()
qpu_result_time = (c_end - c_start)
print("3. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)

3. PROBLEM - QPU ÇÖZÜM SÜRESİ: 7.305860757827759
```

Üçüncü Problem - CPU ile Çözümleme

```
In [78]: c_start = time.time()
cpu_result = dnx.min_vertex_color(mcv3, cpu_sampler)
c_end = time.time()
cpu_result_time = (c_end - c_start)
print("3. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)

3. PROBLEM - CPU ÇÖZÜM SÜRESİ: 0.11782503128051758
```

Birinci Problem - Çözümlerin Karşılaştırılması

```
In [79]: print("1. PROBLEM - QPU SONUÇ: ", qpu_result_a)
print("1. PROBLEM - CPU SONUÇ: ", cpu_result_a)

1. PROBLEM - QPU SONUÇ: {0: 0, 1: 1, 2: 2}
1. PROBLEM - CPU SONUÇ: {0: 2, 1: 0, 2: 1}
```

İkinci Problem - Çözümlerin Karşılaştırılması

```
In [603]: print("2. PROBLEM - QPU SONUÇ: ", qpu_result_b)
print("2. PROBLEM - CPU SONUÇ: ", cpu_result_b)

2. PROBLEM - QPU SONUÇ: {0: 2, 1: 3, 2: 0, 3: 1, 4: 4}
2. PROBLEM - CPU SONUÇ: {0: 3, 1: 4, 2: 1, 3: 2, 4: 0}
```

Üçüncü Problem - Çözümlerin Karşılaştırılması

04/02/2021

TEZ_DOC

```
In [81]: print("3. PROBLEM - QPU SONUÇ: ", qpu_result)
print("3. PROBLEM - CPU SONUÇ: ", cpu_result)

1. PROBLEM - QPU SONUÇ: {0: 2, 2: 0, 3: 2}
1. PROBLEM - CPU SONUÇ: {0: 1, 1: 2, 2: 0, 3: 3}
```

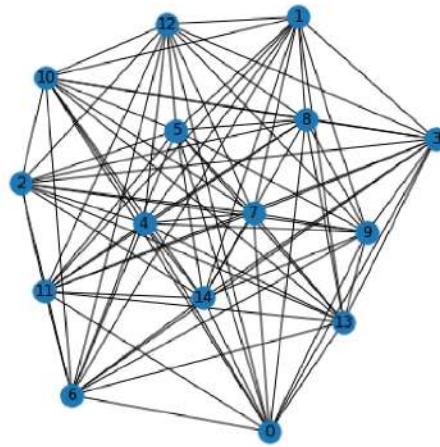
Maximum Clique Problemi

Birinci Problem

```
In [121]: mxcl = nx.turan_graph(15, 5)

for (u,v,w) in mxcl.edges(data=True):
    w['weight'] = random.randint(0,20)

plt.figure(figsize=(5,5))
nx.draw(mxcl, with_labels=True, font_weights='bold')
```



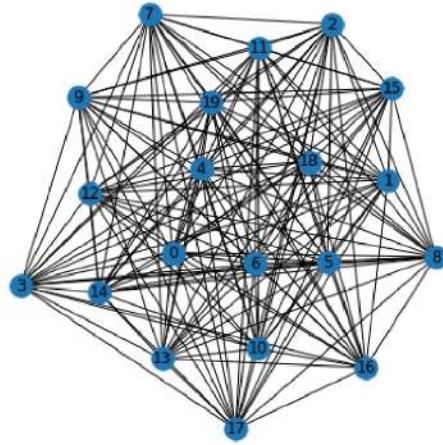
İkinci Problem

04/02/2021 TEZ_DOC

```
In [125]: mxc2 = nx.turan_graph(20, 8)

for (u,v,w) in mxc2.edges(data=True):
    w['weight'] = random.randint(0,20)

plt.figure(figsize=(5,5))
nx.draw(mxc2, with_labels=True, font_weights='bold')
```



Üçüncü Problem

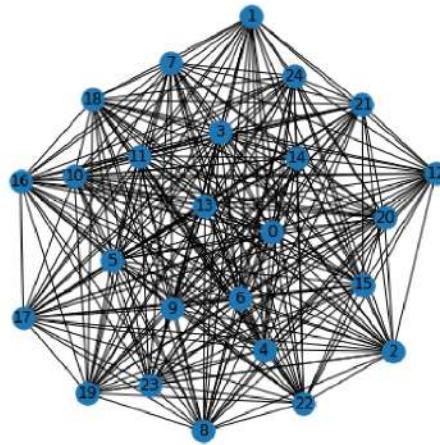
04/02/2021

TEZ_DOC

```
In [126]: mxc3 = nx.turan_graph(25, 13)

for (u,v,w) in mxc3.edges(data=True):
    w['weight'] = random.randint(0,20)

plt.figure(figsize=(5,5))
nx.draw(mxc3, with_labels=True, font_weights='bold')
```



Birinci Problem - QPU ile Çözümleme

```
In [604]: c_start = time.time()

qpu_result_a = dnx.maximum_clique(mxc1, qpu_sampler)

c_end = time.time()

qpu_result_time = (c_end - c_start)

print("1. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)
```

1. PROBLEM - QPU ÇÖZÜM SÜRESİ: 4.724148988723755

Birinci Problem - CPU ile Çözümleme

localhost:8888/nbconvert/html/Macintosh HD/Users/egedursun/Desktop/Tez/TEZ_DOC.ipynb?download=false

34/146

04/02/2021

TEZ_DOC

```
In [605]: c_start = time.time()
cpu_result_a = dnx.maximum_clique(mxc1, cpu_sampler)
c_end = time.time()
cpu_result_time = (c_end - c_start)
print("1. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)
```

1. PROBLEM - CPU ÇÖZÜM SÜRESİ: 0.06234598159790039

İkinci Problem - QPU ile Çözümleme

```
In [606]: c_start = time.time()
qpu_result_b = dnx.maximum_clique(mxc2, qpu_sampler)
c_end = time.time()
qpu_result_time = (c_end - c_start)
print("2. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)
```

2. PROBLEM - QPU ÇÖZÜM SÜRESİ: 4.968411922454834

İkinci Problem - CPU ile Çözümleme

```
In [607]: c_start = time.time()
cpu_result_b = dnx.maximum_clique(mxc2, cpu_sampler)
c_end = time.time()
cpu_result_time = (c_end - c_start)
print("2. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)
```

2. PROBLEM - CPU ÇÖZÜM SÜRESİ: 1.8269367218017578

Üçüncü Problem - QPU ile Çözümleme

04/02/2021

TEZ_DOC

```
In [608]: c_start = time.time()
qpu_result = dnx.maximum_clique(mxc3, qpu_sampler)
c_end = time.time()
qpu_result_time = (c_end - c_start)
print("3. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)

3. PROBLEM - QPU ÇÖZÜM SÜRESİ: 3.9754130840301514
```

Üçüncü Problem - CPU ile Çözümleme

```
In [612]: c_start = time.time()
cpu_result = dnx.maximum_clique(mxc3, cpu_sampler)
c_end = time.time()
cpu_result_time = (c_end - c_start)
print("3. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)

3. PROBLEM - CPU ÇÖZÜM SÜRESİ: 57.896332025527954
```

Birinci Problem - Çözümlerin Karşılaştırılması

```
In [609]: print("1. PROBLEM - QPU SONUÇ: ", qpu_result_a)
print("1. PROBLEM - CPU SONUÇ: ", cpu_result_a)

1. PROBLEM - QPU SONUÇ: [1, 5, 8, 10, 14]
1. PROBLEM - CPU SONUÇ: [2, 3, 7, 9, 12]
```

İkinci Problem - Çözümlerin Karşılaştırılması

```
In [610]: print("2. PROBLEM - QPU SONUÇ: ", qpu_result_b)
print("2. PROBLEM - CPU SONUÇ: ", cpu_result_b)

2. PROBLEM - QPU SONUÇ: [0, 2, 4, 6, 10, 12, 14, 18]
2. PROBLEM - CPU SONUÇ: [1, 3, 4, 7, 10, 11, 16, 18]
```

Üçüncü Problem - Çözümlerin Karşılaştırılması

04/02/2021

TEZ_DOC

```
In [613]: print("3. PROBLEM - QPU SONUÇ: ", qpu_result)
print("3. PROBLEM - CPU SONUÇ: ", cpu_result)

3. PROBLEM - QPU SONUÇ: [0, 2, 3, 5, 8, 9, 11, 13, 16, 18, 20, 22, 23]
3. PROBLEM - CPU SONUÇ: [0, 2, 4, 5, 7, 10, 11, 13, 16, 17, 20, 22, 2
3]
```

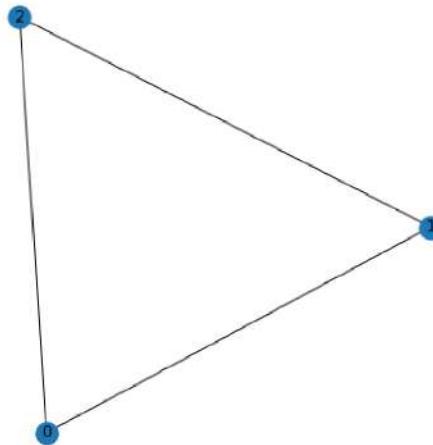
Minimum Maximal Matching Problemı

Birinci Problem

```
In [156]: mmml = nx.complete_graph(3)

for (u,v,w) in mmml.edges(data=True):
    w['weight'] = random.randint(0,20)

plt.figure(figsize=(5,5))
nx.draw(mmml, with_labels=True, font_weights='bold')
```



İkinci Problem

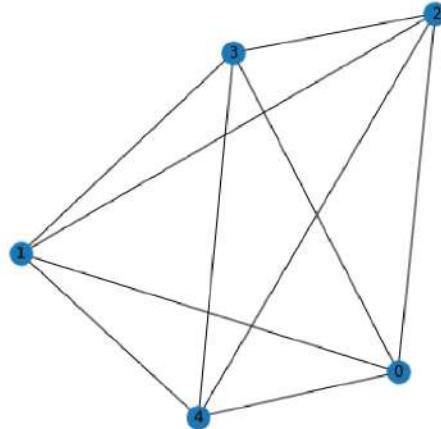
04/02/2021

TEZ_DOC

```
In [157]: mmm2 = nx.complete_graph(5)

for (u,v,w) in mmm2.edges(data=True):
    w['weight'] = random.randint(0,20)

plt.figure(figsize=(5,5))
nx.draw(mmm2, with_labels=True, font_weights='bold')
```



Üçüncü Problem

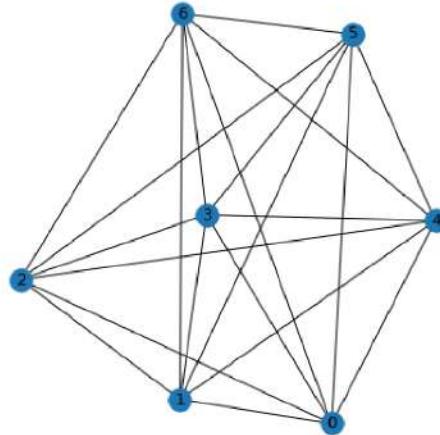
04/02/2021

TEZ_DOC

```
In [158]: mmm3 = nx.complete_graph(7)

for (u,v,w) in mmm3.edges(data=True):
    w['weight'] = random.randint(0,20)

plt.figure(figsize=(5,5))
nx.draw(mmm3, with_labels=True, font_weights='bold')
```



Birinci Problem - QPU ile Çözümleme

```
In [159]: c_start = time.time()

qpu_result_a = dnx.min_maximal_matching(mmm1, qpu_sampler)

c_end = time.time()

qpu_result_time = (c_end - c_start)

print("1. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)
```

1. PROBLEM - QPU ÇÖZÜM SÜRESİ: 5.174998998641968

Birinci Problem - CPU ile Çözümleme

localhost:8888/nbconvert/html/Macintosh HD/Users/egedursun/Desktop/Tez/TEZ_DOC.ipynb?download=false

39/146

04/02/2021

TEZ_DOC

```
In [160]: c_start = time.time()
cpu_result_a = dnx.min_maximal_matching(mmm1, cpu_sampler)
c_end = time.time()
cpu_result_time = (c_end - c_start)
print("1. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)
```

1. PROBLEM - CPU ÇÖZÜM SÜRESİ: 0.0010380744934082031

İkinci Problem - QPU ile Çözümleme

```
In [161]: c_start = time.time()
qpu_result_b = dnx.min_maximal_matching(mmm2, qpu_sampler)
c_end = time.time()
qpu_result_time = (c_end - c_start)
print("2. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)
```

2. PROBLEM - QPU ÇÖZÜM SÜRESİ: 6.901879787445068

İkinci Problem - CPU ile Çözümleme

```
In [162]: c_start = time.time()
cpu_result_b = dnx.min_maximal_matching(mmm2, cpu_sampler)
c_end = time.time()
cpu_result_time = (c_end - c_start)
print("2. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)
```

2. PROBLEM - CPU ÇÖZÜM SÜRESİ: 0.006217002868652344

Üçüncü Problem - QPU ile Çözümleme

04/02/2021

TEZ_DOC

```
In [163]: c_start = time.time()

qpu_result = dnx.min_maximal_matching(mmm3, qpu_sampler)

c_end = time.time()

qpu_result_time = (c_end - c_start)

print("3. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)
```

3. PROBLEM - QPU ÇÖZÜM SÜRESİ: 5.537613153457642

Üçüncü Problem - CPU ile Çözümleme

```
In [164]: c_start = time.time()

cpu_result = dnx.min_maximal_matching(mmm3, cpu_sampler)

c_end = time.time()

cpu_result_time = (c_end - c_start)

print("3. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)
```

3. PROBLEM - CPU ÇÖZÜM SÜRESİ: 9.340500116348267

Birinci Problem - Çözümlerin Karşılaştırılması

```
In [165]: print("1. PROBLEM - QPU SONUÇ: ", qpu_result_a)
print("1. PROBLEM - CPU SONUÇ: ", cpu_result_a)
```

1. PROBLEM - QPU SONUÇ: {(0, 2)}
 1. PROBLEM - CPU SONUÇ: {(0, 1)}

İkinci Problem - Çözümlerin Karşılaştırılması

```
In [166]: print("2. PROBLEM - QPU SONUÇ: ", qpu_result_b)
print("2. PROBLEM - CPU SONUÇ: ", cpu_result_b)
```

2. PROBLEM - QPU SONUÇ: {(0, 1), (1, 3), (0, 2), (2, 4)}
 2. PROBLEM - CPU SONUÇ: {(1, 2), (0, 3)}

Üçüncü Problem - Çözümlerin Karşılaştırılması

04/02/2021

TEZ_DOC

```
In [167]: print("3. PROBLEM - QPU SONUÇ: ", qpu_result)
print("3. PROBLEM - CPU SONUÇ: ", cpu_result)

3. PROBLEM - QPU SONUÇ:  {(1, 5), (0, 2), (1, 4), (0, 4)}
3. PROBLEM - CPU SONUÇ:  {(1, 5), (4, 6), (0, 2)}
```

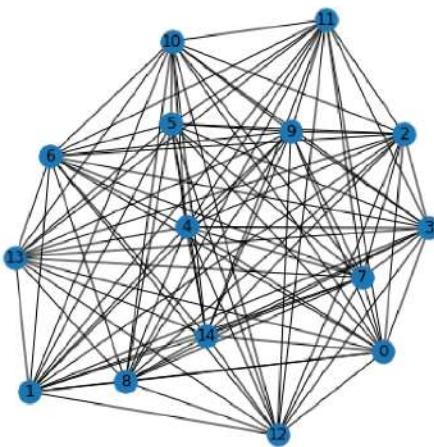
Maximum Cut Problemı

Birinci Problem

```
In [194]: mxcut1 = nx.complete_graph(15)

for (u,v,w) in mxcut1.edges(data=True):
    w['weight'] = random.randint(0,20)

plt.figure(figsize=(5,5))
nx.draw(mxcut1, with_labels=True, font_weights='bold')
```



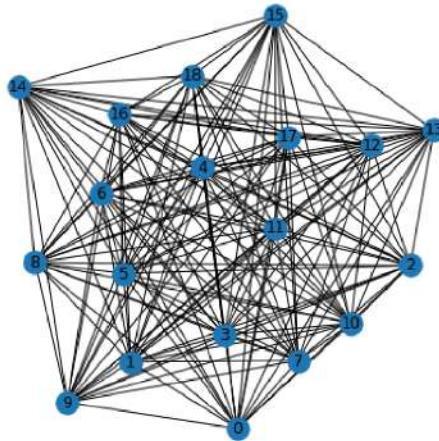
İkinci Problem

04/02/2021 TEZ_DOC

```
In [195]: mxcut2 = nx.complete_graph(19)

for (u,v,w) in mxcut2.edges(data=True):
    w['weight'] = random.randint(0,20)

plt.figure(figsize=(5,5))
nx.draw(mxcut2, with_labels=True, font_weights='bold')
```



Üçüncü Problem

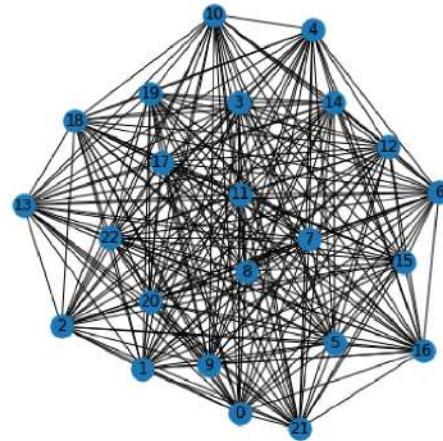
04/02/2021

TEZ_DOC

```
In [196]: mxcut3 = nx.complete_graph(23)

for (u,v,w) in mxcut3.edges(data=True):
    w['weight'] = random.randint(0,20)

plt.figure(figsize=(5,5))
nx.draw(mxcut3, with_labels=True, font_weights='bold')
```



Birinci Problem - QPU ile Çözümleme

```
In [197]: c_start = time.time()

qpu_result_a = dnx.maximum_cut(mxcut1, qpu_sampler)

c_end = time.time()

qpu_result_time = (c_end - c_start)

print("1. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)
```

1. PROBLEM - QPU ÇÖZÜM SÜRESİ: 5.048703193664551

Birinci Problem - CPU ile Çözümleme

04/02/2021

TEZ_DOC

```
In [198]: c_start = time.time()
cpu_result_a = dnx.maximum_cut(mxcut1, cpu_sampler)
c_end = time.time()
cpu_result_time = (c_end - c_start)
print("1. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)
```

1. PROBLEM - CPU ÇÖZÜM SÜRESİ: 0.09636402130126953

İkinci Problem - QPU ile Çözümleme

```
In [199]: c_start = time.time()
qpu_result_b = dnx.maximum_cut(mxcut2, qpu_sampler)
c_end = time.time()
qpu_result_time = (c_end - c_start)
print("2. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)
```

2. PROBLEM - QPU ÇÖZÜM SÜRESİ: 7.465651988983154

İkinci Problem - CPU ile Çözümleme

```
In [200]: c_start = time.time()
cpu_result_b = dnx.maximum_cut(mxcut2, cpu_sampler)
c_end = time.time()
cpu_result_time = (c_end - c_start)
print("2. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)
```

2. PROBLEM - CPU ÇÖZÜM SÜRESİ: 1.466797113418579

Üçüncü Problem - QPU ile Çözümleme

04/02/2021

TEZ_DOC

```
In [201]: c_start = time.time()
qpu_result = dnx.maximum_cut(mxcut3, qpu_sampler)
c_end = time.time()
qpu_result_time = (c_end - c_start)
print("3. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)

3. PROBLEM - QPU ÇÖZÜM SÜRESİ: 4.111576080322266
```

Üçüncü Problem - CPU ile Çözümleme

```
In [202]: c_start = time.time()
cpu_result = dnx.maximum_cut(mxcut3, cpu_sampler)
c_end = time.time()
cpu_result_time = (c_end - c_start)
print("3. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)

3. PROBLEM - CPU ÇÖZÜM SÜRESİ: 46.15278220176697
```

Birinci Problem - Çözümlerin Karşılaştırılması

```
In [203]: print("1. PROBLEM - QPU SONUÇ: ", qpu_result_a)
print("1. PROBLEM - CPU SONUÇ: ", cpu_result_a)

1. PROBLEM - QPU SONUÇ: {1, 4, 6, 10, 11, 12, 14}
1. PROBLEM - CPU SONUÇ: {0, 3, 4, 5, 7, 10, 13}
```

İkinci Problem - Çözümlerin Karşılaştırılması

```
In [204]: print("2. PROBLEM - QPU SONUÇ: ", qpu_result_b)
print("2. PROBLEM - CPU SONUÇ: ", cpu_result_b)

2. PROBLEM - QPU SONUÇ: {0, 1, 4, 6, 7, 8, 14, 15}
2. PROBLEM - CPU SONUÇ: {1, 2, 3, 4, 10, 11, 13, 15, 16, 17}
```

Üçüncü Problem - Çözümlerin Karşılaştırılması

04/02/2021

TEZ_DOC

```
In [205]: print("3. PROBLEM - QPU SONUÇ: ", qpu_result)
print("3. PROBLEM - CPU SONUÇ: ", cpu_result)

3. PROBLEM - QPU SONUÇ: {0, 1, 4, 6, 7, 8, 11, 15, 17, 18, 19, 22}
3. PROBLEM - CPU SONUÇ: {0, 5, 7, 10, 12, 13, 15, 18, 19, 20, 21}
```

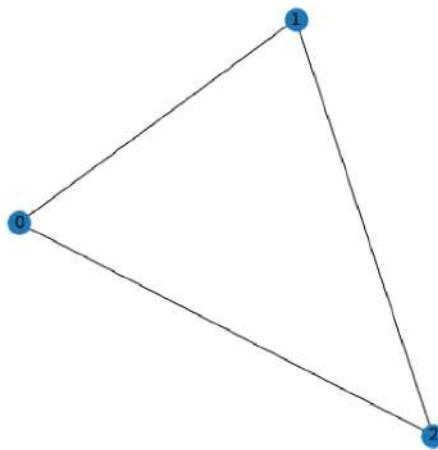
Traveling Salesperson Problemı

Birinci Problem

```
In [217]: tspl = nx.complete_graph(3)

for (u,v,w) in tspl.edges(data=True):
    w['weight'] = random.randint(10,50)

plt.figure(figsize=(5,5))
nx.draw(tspl, with_labels=True, font_weights='bold')
```



İkinci Problem

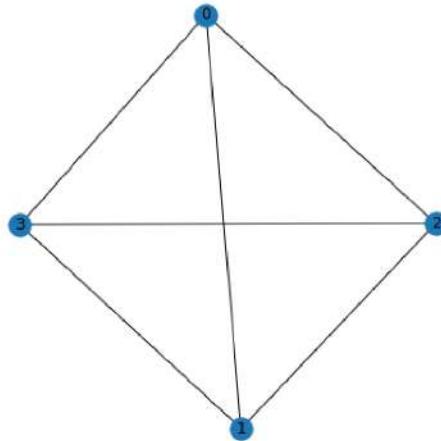
04/02/2021

TEZ_DOC

```
In [219]: tsp2 = nx.complete_graph(4)

for (u,v,w) in tsp2.edges(data=True):
    w['weight'] = random.randint(10,50)

plt.figure(figsize=(5,5))
nx.draw(tsp2, with_labels=True, font_weights='bold')
```



Üçüncü Problem

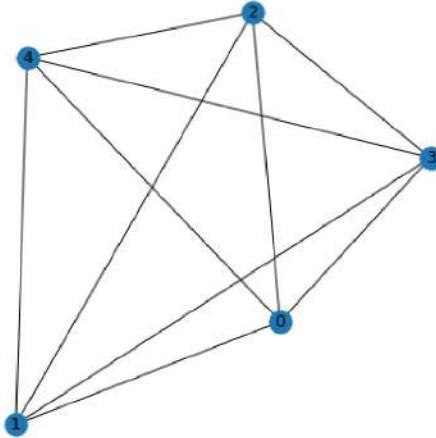
04/02/2021

TEZ_DOC

```
In [220]: tsp3 = nx.complete_graph(5)

for (u,v,w) in tsp3.edges(data=True):
    w['weight'] = random.randint(10,50)

plt.figure(figsize=(5,5))
nx.draw(tsp3, with_labels=True, font_weights='bold')
```



Birinci Problem - QPU ile Çözümleme

```
In [221]: c_start = time.time()

qpu_result_a = dnx.traveling_salesperson(tspl, qpu_sampler)

c_end = time.time()

qpu_result_time = (c_end - c_start)

print("1. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)
```

1. PROBLEM - QPU ÇÖZÜM SÜRESİ: 5.453104019165039

Birinci Problem - CPU ile Çözümleme

04/02/2021

TEZ_DOC

```
In [222]: c_start = time.time()
cpu_result_a = dnx.traveling_salesperson(tsp1, cpu_sampler)
c_end = time.time()
cpu_result_time = (c_end - c_start)
print("1. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)
```

1. PROBLEM - CPU ÇÖZÜM SÜRESİ: 0.003008127212524414

İkinci Problem - QPU ile Çözümleme

```
In [223]: c_start = time.time()
qpu_result_b = dnx.traveling_salesperson(tsp2, qpu_sampler)
c_end = time.time()
qpu_result_time = (c_end - c_start)
print("2. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)
```

2. PROBLEM - QPU ÇÖZÜM SÜRESİ: 6.469345808029175

İkinci Problem - CPU ile Çözümleme

```
In [224]: c_start = time.time()
cpu_result_b = dnx.traveling_salesperson(tsp2, cpu_sampler)
c_end = time.time()
cpu_result_time = (c_end - c_start)
print("2. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)
```

2. PROBLEM - CPU ÇÖZÜM SÜRESİ: 0.15728998184204102

Üçüncü Problem - QPU ile Çözümleme

04/02/2021

TEZ_DOC

```
In [225]: c_start = time.time()
qpu_result = dnx.traveling_salesperson(tsp3, qpu_sampler)
c_end = time.time()
qpu_result_time = (c_end - c_start)
print("3. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)

3. PROBLEM - QPU ÇÖZÜM SÜRESİ:  8.27685022354126
```

Üçüncü Problem - CPU ile Çözümleme

```
In [226]: c_start = time.time()
cpu_result = dnx.traveling_salesperson(tsp3, cpu_sampler)
c_end = time.time()
cpu_result_time = (c_end - c_start)
print("3. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)

3. PROBLEM - CPU ÇÖZÜM SÜRESİ:  184.03486275672913
```

Birinci Problem - Çözümlerin Karşılaştırılması

```
In [227]: print("1. PROBLEM - QPU SONUÇ: ", qpu_result_a)
print("1. PROBLEM - CPU SONUÇ: ", cpu_result_a)

1. PROBLEM - QPU SONUÇ:  [1, 1, 1]
1. PROBLEM - CPU SONUÇ:  [0, 1, 2]
```

İkinci Problem - Çözümlerin Karşılaştırılması

```
In [228]: print("2. PROBLEM - QPU SONUÇ: ", qpu_result_b)
print("2. PROBLEM - CPU SONUÇ: ", cpu_result_b)

2. PROBLEM - QPU SONUÇ:  [None, None, None, 3]
2. PROBLEM - CPU SONUÇ:  [3, 1, 0, 2]
```

Üçüncü Problem - Çözümlerin Karşılaştırılması

04/02/2021 TEZ_DOC

```
In [229]: print("3. PROBLEM - QPU SONUÇ: ", qpu_result)
print("3. PROBLEM - CPU SONUÇ: ", cpu_result)

3. PROBLEM - QPU SONUÇ:  [1, 2, 1, 2, None]
3. PROBLEM - CPU SONUÇ:  [0, 1, 2, 3, 4]
```

Structural Imbalance Problemı

Birinci Problem

```
In [261]: sim1 = nx.complete_graph(5)
sim1.add_edges_from([(u, v, {'sign': 2*random.randint(0, 1) - 1}) for u,
v in sim1.edges])

nx.relabel_nodes(sim1,
                 {0: 'Ayse', 1: 'Baris', 2: 'Ceren', 3: 'Deniz', 4:'Eda'}
                ),
copy=False)

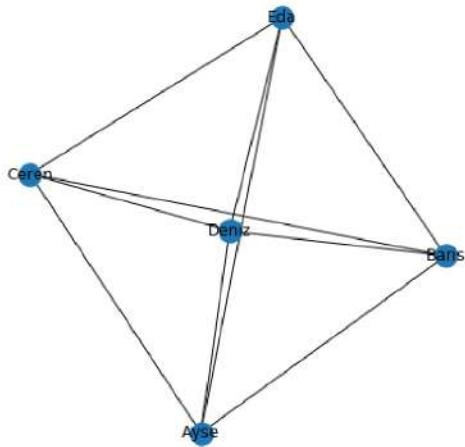
print('Pozitif (Arkadaşcil) ilişkiler: \n\t' + '\n\t'.join(
    list(x + " & " + y for (x, y, sign) in sim1.edges(data='sign') if (sign == 1)))
)
print('Negatif (Düşmancıl) ilişkiler: \n\t' + '\n\t'.join(
    list(x + " & " + y for (x, y, sign) in sim1.edges(data='sign') if (sign == -1)))

Pozitif (Arkadaşcil) ilişkiler:
Ayse & Ceren
Baris & Ceren
Ceren & Deniz
Negatif (Düşmancıl) ilişkiler:
Ayse & Baris
Ayse & Deniz
Ayse & Eda
Baris & Deniz
Baris & Eda
Ceren & Eda
Deniz & Eda
```

04/02/2021

TEZ_DOC

```
In [262]: plt.figure(figsize=(5,5))
nx.draw(sim1, with_labels=True, font_weights='bold')
```

**İkinci Problem**

04/02/2021

TEZ_DOC

```
In [263]: sim2 = nx.complete_graph(15)
sim2.add_edges_from([(u, v, {'sign': 2*random.randint(0, 1) - 1}) for u,
v in sim2.edges])

nx.relabel_nodes(sim2,
                 {0: 'Ayse', 1: 'Baris', 2: 'Ceren', 3: 'Deniz', 4: 'Eda',
                  5: 'Fahrettin', 6: 'Gaye',
                  7: 'Hulya', 8: 'Idil', 9: 'Jale', 10: 'Kadir', 11: 'Lale',
                  12: 'Melis', 13: 'Nedim', 14: 'Onur'},
                 copy=False)

print('Pozitif (Arkadaşçıł) ilişkiler: \n\t' + '\n\t'.join(
    list(x + " & " + y for (x, y, sign) in sim2.edges(data='sign') if (sign == 1))))
print('Negatif (Düşmancıl) ilişkiler: \n\t' + '\n\t'.join(
    list(x + " & " + y for (x, y, sign) in sim2.edges(data='sign') if (sign == -1))))
```

Pozitif (Arkadaşcil) ilişkiler:

Ayse & Baris
Ayse & Deniz
Ayse & Eda
Ayse & Gaye
Ayse & Hulya
Ayse & Melis
Ayse & Nedim
Baris & Ceren
Baris & Deniz
Baris & Fahrettin
Baris & Gaye
Baris & Hulya
Baris & Melis
Baris & Onur
Ceren & Deniz
Ceren & Fahrettin
Ceren & Hulya
Ceren & İdil
Ceren & Kadir
Ceren & Lale
Ceren & Nedim
Ceren & Onur
Deniz & Hulya
Deniz & Jale
Deniz & Nedim
Deniz & Onur
Eda & Fahrettin
Eda & Hulya
Eda & İdil
Eda & Lale
Eda & Melis
Fahrettin & Gaye
Gaye & Hulya
Gaye & İdil
Gaye & Lale
Gaye & Onur
Hulya & İdil
Hulya & Jale
Hulya & Lale
Hulya & Melis
Hulya & Onur
İdil & Jale
İdil & Melis
Jale & Kadir
Jale & Melis
Jale & Nedim
Kadir & Lale
Lale & Melis
Lale & Nedim
Melis & Onur
Nedim & Onur

Negatif (Düşmancıl) ilişkiler:

Ayse & Ceren
Ayse & Fahrettin
Ayse & İdil
Ayse & Jale

04/02/2021

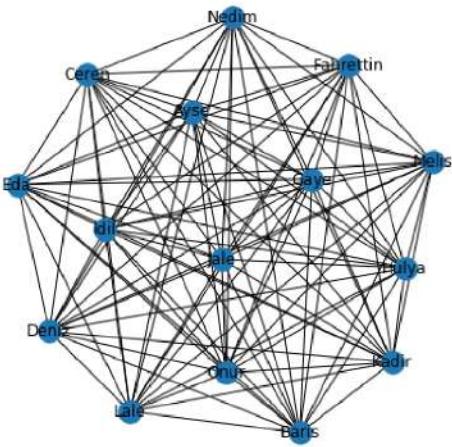
TEZ_DOC

Ayse & Kadir
Ayse & Lale
Ayse & Onur
Baris & Eda
Baris & idil
Baris & Jale
Baris & Kadir
Baris & Lale
Baris & Nedim
Ceren & Eda
Ceren & Gaye
Ceren & Jale
Ceren & Melis
Deniz & Eda
Deniz & Fahrettin
Deniz & Gaye
Deniz & idil
Deniz & Kadir
Deniz & Lale
Deniz & Melis
Eda & Gaye
Eda & Jale
Eda & Kadir
Eda & Nedim
Eda & Onur
Fahrettin & Hulya
Fahrettin & idil
Fahrettin & Jale
Fahrettin & Kadir
Fahrettin & Lale
Fahrettin & Melis
Fahrettin & Nedim
Fahrettin & Onur
Gaye & Jale
Gaye & Kadir
Gaye & Melis
Gaye & Nedim
Hulya & Kadir
Hulya & Nedim
idil & Kadir
idil & Lale
idil & Nedim
idil & Onur
Jale & Lale
Jale & Onur
Kadir & Melis
Kadir & Nedim
Kadir & Onur
Lale & Onur
Melis & Nedim

04/02/2021

TEZ_DOC

```
In [264]: plt.figure(figsize=(5,5))
nx.draw(sim2, with_labels=True, font_weights='bold')
```



Üçüncü Problem

04/02/2021

TEZ_DOC

```
In [265]: sim3 = nx.complete_graph(25)
sim3.add_edges_from([(u, v, {'sign': 2*random.randint(0, 1) - 1}) for u,
v in sim3.edges])

nx.relabel_nodes(sim3,
                 {0: 'Ayse', 1: 'Baris', 2: 'Ceren', 3: 'Deniz', 4: 'Eda',
                  5: 'Fahrettin', 6: 'Gaye',
                  7: 'Hulya', 8: 'Idil', 9: 'Jale', 10: 'Kadir', 11: 'Lale',
                  12: 'Melis', 13: 'Nedim', 14: 'Onur',
                  15: 'Pelin', 16: 'Rasim', 17: 'Selin', 18: 'Tolga', 19: 'Ulvi',
                  20: 'Volkan', 21: 'Yavuz',
                  22: 'Zeliha', 23: 'Anil', 24: 'Beril'},
                 copy=False)

print('Pozitif (Arkadaşcil) ilişkiler: \n\t' + '\n\t'.join(
    list(x + " & " + y for (x, y, sign) in sim3.edges(data='sign') if (sign == 1))))
print('Negatif (Düşmancıl) ilişkiler: \n\t' + '\n\t'.join(
    list(x + " & " + y for (x, y, sign) in sim3.edges(data='sign') if (sign == -1))))
```

Pozitif (Arkadaşcil) ilişkiler:

Ayse & Eda
Ayse & Fahrettin
Ayse & Hulya
Ayse & İdil
Ayse & Jale
Ayse & Kadir
Ayse & Lale
Ayse & Nedim
Ayse & Pelin
Ayse & Ulvi
Ayse & Volkan
Ayse & Anıl
Ayse & Beril
Baris & Ceren
Baris & Eda
Baris & İdil
Baris & Lale
Baris & Nedim
Baris & Pelin
Baris & Rasim
Baris & Tolga
Baris & Ulvi
Baris & Yavuz
Baris & Zeliha
Baris & Anıl
Baris & Beril
Ceren & Deniz
Ceren & Fahrettin
Ceren & Gaye
Ceren & Hulya
Ceren & İdil
Ceren & Jale
Ceren & Kadir
Ceren & Lale
Ceren & Onur
Ceren & Rasim
Ceren & Zeliha
Ceren & Beril
Deniz & İdil
Deniz & Onur
Deniz & Rasim
Deniz & Selin
Deniz & Ulvi
Deniz & Volkan
Deniz & Anıl
Deniz & Beril
Eda & Fahrettin
Eda & İdil
Eda & Jale
Eda & Kadir
Eda & Lale
Eda & Onur
Eda & Pelin
Eda & Rasim
Eda & Ulvi
Eda & Volkan

Eda & Yavuz
Eda & Zeliha
Eda & Anil
Fahrettin & Gaye
Fahrettin & Lale
Fahrettin & Pelin
Fahrettin & Rasim
Fahrettin & Selin
Fahrettin & Ulvi
Fahrettin & Yavuz
Fahrettin & Beril
Gaye & Melis
Gaye & Rasim
Gaye & Selin
Gaye & Volkhan
Gaye & Yavuz
Gaye & Beril
Hulya & Nedim
Hulya & Onur
Hulya & Selin
Hulya & Volkhan
Hulya & Yavuz
Hulya & Beril
İdil & Kadir
İdil & Lale
İdil & Nedim
İdil & Rasim
İdil & Ulvi
İdil & Volkhan
İdil & Zeliha
İdil & Beril
Jale & Kadir
Jale & Nedim
Jale & Onur
Jale & Pelin
Jale & Rasim
Jale & Selin
Jale & Volkhan
Jale & Yavuz
Jale & Zeliha
Kadir & Lale
Kadir & Onur
Kadir & Selin
Kadir & Tolga
Kadir & Ulvi
Kadir & Volkhan
Kadir & Yavuz
Kadir & Zeliha
Lale & Nedim
Lale & Onur
Lale & Pelin
Lale & Rasim
Lale & Selin
Lale & Tolga
Lale & Zeliha
Melis & Nedim
Melis & Pelin

04/02/2021

TEZ_DOC

Melis & Volkan
Melis & Yavuz
Nedim & Pelin
Nedim & Selin
Nedim & Tolga
Nedim & Ulvi
Nedim & Volkan
Onur & Pelin
Onur & Tolga
Onur & Ulvi
Onur & Volkan
Onur & Yavuz
Onur & Anil
Onur & Beril
Pelin & Rasim
Pelin & Ulvi
Pelin & Beril
Rasim & Selin
Rasim & Tolga
Rasim & Zeliha
Rasim & Anil
Selin & Tolga
Selin & Ulvi
Selin & Zeliha
Tolga & Ulvi
Tolga & Beril
Ulvi & Volkan
Ulvi & Anil
Ulvi & Beril
Volkan & Yavuz
Volkan & Beril
Yavuz & Zeliha
Zeliha & Anil
Negatif (Düşmancıl) ilişkiler:
Ayşe & Barış
Ayşe & Ceren
Ayşe & Deniz
Ayşe & Gaye
Ayşe & Melis
Ayşe & Onur
Ayşe & Rasim
Ayşe & Selin
Ayşe & Tolga
Ayşe & Yavuz
Ayşe & Zeliha
Barış & Deniz
Barış & Fahrettin
Barış & Gaye
Barış & Hulya
Barış & Jale
Barış & Kadir
Barış & Melis
Barış & Onur
Barış & Selin
Barış & Volkan
Ceren & Eda
Ceren & Melis

04/02/2021

TEZ_DOC

Ceren & Nedim
Ceren & Pelin
Ceren & Selin
Ceren & Tolga
Ceren & Ulvi
Ceren & Volkan
Ceren & Yavuz
Ceren & Anil
Deniz & Eda
Deniz & Fahrettin
Deniz & Gaye
Deniz & Hulya
Deniz & Jale
Deniz & Kadir
Deniz & Lale
Deniz & Melis
Deniz & Nedim
Deniz & Pelin
Deniz & Tolga
Deniz & Yavuz
Deniz & Zeliha
Eda & Gaye
Eda & Hulya
Eda & Melis
Eda & Nedim
Eda & Selin
Eda & Tolga
Eda & Beril
Fahrettin & Hulya
Fahrettin & idil
Fahrettin & Jale
Fahrettin & Kadir
Fahrettin & Melis
Fahrettin & Nedim
Fahrettin & Onur
Fahrettin & Tolga
Fahrettin & Volkan
Fahrettin & Zeliha
Fahrettin & Anil
Gaye & Hulya
Gaye & idil
Gaye & Jale
Gaye & Kadir
Gaye & Lale
Gaye & Nedim
Gaye & Onur
Gaye & Pelin
Gaye & Tolga
Gaye & Ulvi
Gaye & Zeliha
Gaye & Anil
Hulya & idil
Hulya & Jale
Hulya & Kadir
Hulya & Lale
Hulya & Melis
Hulya & Pelin

localhost:8888/nbconvert/html/Macintosh HD/Users/egedursun/Desktop/Tez/TEZ_DOC.ipynb?download=false

62/146

04/02/2021

TEZ_DOC

Hulya & Rasim
Hulya & Tolga
Hulya & Ulvi
Hulya & Zeliha
Hulya & Anil
İdil & Jale
İdil & Melis
İdil & Onur
İdil & Pelin
İdil & Selin
İdil & Tolga
İdil & Yavuz
İdil & Anil
Jale & Lale
Jale & Melis
Jale & Tolga
Jale & Ulvi
Jale & Anil
Jale & Beril
Kadir & Melis
Kadir & Nedim
Kadir & Pelin
Kadir & Rasim
Kadir & Anil
Kadir & Beril
Lale & Melis
Lale & Ulvi
Lale & Volkan
Lale & Yavuz
Lale & Anil
Lale & Beril
Melis & Onur
Melis & Rasim
Melis & Selin
Melis & Tolga
Melis & Ulvi
Melis & Zeliha
Melis & Anil
Melis & Beril
Nedim & Onur
Nedim & Rasim
Nedim & Yavuz
Nedim & Zeliha
Nedim & Anil
Nedim & Beril
Onur & Rasim
Onur & Selin
Onur & Zeliha
Pelin & Selin
Pelin & Tolga
Pelin & Volkan
Pelin & Yavuz
Pelin & Zeliha
Pelin & Anil
Rasim & Ulvi
Rasim & Volkan
Rasim & Yavuz

localhost:8888/nbconvert/html/Macintosh HD/Users/egedursun/Desktop/Tez/TEZ_DOC.ipynb?download=false

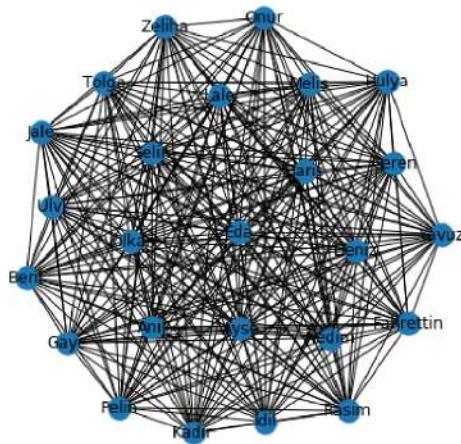
63/146

04/02/2021

TEZ_DOC

```
Rasim & Beril
Selin & Volkan
Selin & Yavuz
Selin & Anil
Selin & Beril
Tolga & Volkan
Tolga & Yavuz
Tolga & Zeliha
Tolga & Anil
Ulvi & Yavuz
Ulvi & Zeliha
Volkan & Zeliha
Volkan & Anil
Yavuz & Anil
Yavuz & Beril
Zeliha & Beril
Anil & Beril
```

```
In [266]: plt.figure(figsize=(5,5))
nx.draw(sim3, with_labels=True, font_weights='bold')
```



Birinci Problem - QPU ile Çözümleme

04/02/2021

TEZ_DOC

```
In [274]: c_start = time.time()

qpu_imbalance_a, qpu_bicoloring_a = dnx.structural_imbalance(sim1, qpu_s
ampler)

c_end = time.time()

qpu_result_time = (c_end - c_start)

print("1. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)
```

1. PROBLEM - QPU ÇÖZÜM SÜRESİ: 5.223376989364624

Birinci Problem - CPU ile Çözümleme

```
In [275]: c_start = time.time()

cpu_imbalance_a, cpu_bicoloring_a = dnx.structural_imbalance(sim1, cpu_s
ampler)

c_end = time.time()

cpu_result_time = (c_end - c_start)

print("1. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)
```

1. PROBLEM - CPU ÇÖZÜM SÜRESİ: 0.0013730525970458984

İkinci Problem - QPU ile Çözümleme

```
In [278]: c_start = time.time()

qpu_imbalance_b, qpu_bicoloring_b = dnx.structural_imbalance(sim2, qpu_s
ampler)

c_end = time.time()

qpu_result_time = (c_end - c_start)

print("2. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)
```

2. PROBLEM - QPU ÇÖZÜM SÜRESİ: 5.52672004699707

İkinci Problem - CPU ile Çözümleme

04/02/2021

TEZ_DOC

```
In [279]: c_start = time.time()

cpu_imbalance_b, cpu_bicoloring_b = dnx.structural_imbalance(sim2, cpu_sampler)

c_end = time.time()

cpu_result_time = (c_end - c_start)

print("2. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)
```

2. PROBLEM - CPU ÇÖZÜM SÜRESİ: 0.08593106269836426

Üçüncü Problem - QPU ile Çözümleme

```
In [282]: c_start = time.time()

qpu_imbalance, qpu_bicoloring = dnx.structural_imbalance(sim3, qpu_sampler)

c_end = time.time()

qpu_result_time = (c_end - c_start)

print("3. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)
```

3. PROBLEM - QPU ÇÖZÜM SÜRESİ: 9.294797897338867

Üçüncü Problem - CPU ile Çözümleme

```
In [283]: c_start = time.time()

cpu_imbalance, cpu_bicoloring = dnx.structural_imbalance(sim3, cpu_sampler)

c_end = time.time()

cpu_result_time = (c_end - c_start)

print("3. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)
```

3. PROBLEM - CPU ÇÖZÜM SÜRESİ: 307.7670600414276

Birinci Problem - Çözümlerin Karşılaştırılması

04/02/2021

TEZ_DOC

```
In [276]: for edge in sim1.edges:
    sim1.edges[edge]['frustrated'] = edge in gpu_imbalance_a
for node in sim1.nodes:
    sim1.nodes[node]['color'] = gpu_bicoloring_a[node]

print("1.PROBLEM (QPU):")
print('GRUP A: \n\t' + '\n\t'.join(list(person for (person, color) in qp_u_bicoloring_a.items() if (color == 0))))
print('GRUP B: \n\t' + '\n\t'.join(list(person for (person, color) in qp_u_bicoloring_a.items() if (color == 1))))
print('DENGESİZ/INSTABİL İLİŞKİLER: \n\t' + '\n\t'.join(list(x + " & " +
y for (x, y) in gpu_imbalance_a.keys())))

1.PROBLEM (QPU):
GRUP A:
    Eda
GRUP B:
    Ayse
    Baris
    Ceren
    Deniz
DENGESİZ/INSTABİL İLİŞKİLER:
    Ayse & Baris
    Ayse & Deniz
    Baris & Deniz

In [277]: for edge in sim1.edges:
    sim1.edges[edge]['frustrated'] = edge in cpu_imbalance_a
for node in sim1.nodes:
    sim1.nodes[node]['color'] = cpu_bicoloring_a[node]

print("1.PROBLEM (CPU):")
print('GRUP A: \n\t' + '\n\t'.join(list(person for (person, color) in cp_u_bicoloring_a.items() if (color == 0))))
print('GRUP B: \n\t' + '\n\t'.join(list(person for (person, color) in cp_u_bicoloring_a.items() if (color == 1))))
print('DENGESİZ/INSTABİL İLİŞKİLER: \n\t' + '\n\t'.join(list(x + " & " +
y for (x, y) in cpu_imbalance_a.keys())))

1.PROBLEM (CPU):
GRUP A:
    Ayse
    Baris
    Ceren
    Deniz
GRUP B:
    Eda
DENGESİZ/INSTABİL İLİŞKİLER:
    Ayse & Baris
    Ayse & Deniz
    Baris & Deniz
```

İkinci Problem - Çözümlerin Karşılaştırılması

localhost:8888/nbconvert/html/Macintosh HD/Users/egedursun/Desktop/Tez/TEZ_DOC.ipynb?download=false

67/146

04/02/2021

TEZ_DOC

```
In [280]: for edge in sim2.edges:
    sim2.edges[edge]['frustrated'] = edge in gpu_imbalance_b
for node in sim2.nodes:
    sim2.nodes[node]['color'] = gpu_bicoloring_b[node]

print("2.PROBLEM (QPU):")
print('GRUP A: \n\t' + '\n\t'.join(list(person for (person, color) in qp_u_bicoloring_b.items() if (color == 0))))
print('GRUP B: \n\t' + '\n\t'.join(list(person for (person, color) in qp_u_bicoloring_b.items() if (color == 1))))
print('DENGESİZ/INSTABİL İLİŞKİLER: \n\t' + '\n\t'.join(list(x + " & " +
y for (x, y) in gpu_imbalance_b.keys()))))
```

2.PROBLEM (QPU):
GRUP A:

Ayse
Baris
Deniz
Fahrettin
Gaye
Hulya
Onur

GRUP B:

Ceren
Eda
Jale
Kadir
Lale
Melis
Nedim
İdil

DENGESİZ/INSTABİL İLİŞKİLER:

Ayse & Eda
Ayse & Fahrettin
Ayse & Melis
Ayse & Nedim
Ayse & Onur
Baris & Ceren
Baris & Melis
Ceren & Deniz
Ceren & Eda
Ceren & Fahrettin
Ceren & Hulya
Ceren & Jale
Ceren & Melis
Ceren & Onur
Deniz & Fahrettin
Deniz & Gaye
Deniz & Jale
Deniz & Nedim
Eda & Fahrettin
Eda & Hulya
Eda & Jale
Eda & Kadir
Eda & Nedim
Fahrettin & Hulya
Fahrettin & Onur
Gaye & İdil
Gaye & Lale
Hulya & İdil
Hulya & Jale
Hulya & Lale
Hulya & Melis
İdil & Kadir
İdil & Lale
İdil & Nedim
Jale & Lale
Kadir & Melis
Kadir & Nedim
Melis & Nedim

04/02/2021

TEZ_DOC

```
In [281]: for edge in sim2.edges:
    sim2.edges[edge]['frustrated'] = edge in cpu_imbalance_b
for node in sim2.nodes:
    sim2.nodes[node]['color'] = cpu_bicoloring_b[node]

print("2.PROBLEM (CPU):")
print('GRUP A: \n\t' + '\n\t'.join(list(person for (person, color) in cp
u_bicoloring_b.items() if (color == 0))))
print('GRUP B: \n\t' + '\n\t'.join(list(person for (person, color) in cp
u_bicoloring_b.items() if (color == 1))))
print('DENGESİZ/INSTABİL İLİŞKİLER: \n\t' + '\n\t'.join(list(x + " & " +
y for (x, y) in cpu_imbalance_b.keys()))))
```

04/02/2021

TEZ_DOC

2.PROBLEM (CPU):
GRUP A:
Ayşe
Barış
Eda
Fahrettin
Gaye
Hulya
Melis
İdil
GRUP B:
Ceren
Deniz
Jale
Kadir
Lale
Nedim
Onur
DENGESİZ/INSTABİL İLİŞKİLER:
Ayşe & Deniz
Ayşe & Fahrettin
Ayşe & İdil
Ayşe & Nedim
Barış & Ceren
Barış & Deniz
Barış & Eda
Barış & İdil
Barış & Onur
Ceren & Fahrettin
Ceren & Hulya
Ceren & İdil
Ceren & Jale
Deniz & Hulya
Deniz & Kadir
Deniz & Lale
Eda & Gaye
Eda & Lale
Fahrettin & Hulya
Fahrettin & İdil
Fahrettin & Melis
Gaye & Lale
Gaye & Melis
Gaye & Onur
Hulya & Jale
Hulya & Lale
Hulya & Onur
İdil & Jale
Jale & Lale
Jale & Melis
Jale & Onur
Kadir & Nedim
Kadir & Onur
Lale & Melis
Lale & Onur
Melis & Onur

Üçüncü Problem - Çözümlerin Karesi İstirilmesi

04/02/2021

TEZ_DOC

```
In [284]: for edge in sim3.edges:
    sim3.edges[edge]['frustrated'] = edge in qpu_imbalance
for node in sim3.nodes:
    sim3.nodes[node]['color'] = qpu_bicoloring[node]

print("3.PROBLEM (QPU):")
print('GRUP A: \n\t' + '\n\t'.join(list(person for (person, color) in qp
u_bicoloring.items() if (color == 0))))
print('GRUP B: \n\t' + '\n\t'.join(list(person for (person, color) in qp
u_bicoloring.items() if (color == 1))))
print('DENGESİZ/INSTABİL İLİŞKİLER: \n\t' + '\n\t'.join(list(x + " & "
y for (x, y) in qpu_imbalance.keys()))))
```

3.PROBLEM (QPU):
GRUP A:

Anıl
Ayşe
Beril
Eda
Hulya
Jale
Lale
Melis
Nedim
Onur
Volkan
Yavuz
İdil

GRUP B:

Barış
Ceren
Deniz
Fahrettin
Gaye
Kadir
Pelin
Rasim
Selin
Tolga
Ulvi
Zeliha

DENGESİZ/INSTABİL İLİŞKİLER:

Ayşe & Fahrettin
Ayşe & Kadir
Ayşe & Melis
Ayşe & Onur
Ayşe & Pelin
Ayşe & Ulvi
Ayşe & Yavuz
Barış & Deniz
Barış & Eda
Barış & Fahrettin
Barış & Gaye
Barış & İdil
Barış & Kadir
Barış & Lale
Barış & Nedim
Barış & Selin
Barış & Yavuz
Barış & Anıl
Barış & Beril
Ceren & Hulya
Ceren & İdil
Ceren & Jale
Ceren & Lale
Ceren & Onur
Ceren & Pelin
Ceren & Selin
Ceren & Tolga
Ceren & Ulvi

Ceren & Beril
Deniz & Fahrettin
Deniz & Gaye
Deniz & idil
Deniz & Kadir
Deniz & Onur
Deniz & Pelin
Deniz & Tolga
Deniz & Volkhan
Deniz & Zeliha
Deniz & Anil
Deniz & Beril
Eda & Fahrettin
Eda & Hulya
Eda & Kadir
Eda & Melis
Eda & Nedim
Eda & Pelin
Eda & Rasim
Eda & Ulvi
Eda & Zeliha
Eda & Beril
Fahrettin & Kadir
Fahrettin & Lale
Fahrettin & Tolga
Fahrettin & Yavuz
Fahrettin & Zeliha
Fahrettin & Beril
Gaye & Kadir
Gaye & Melis
Gaye & Pelin
Gaye & Tolga
Gaye & Ulvi
Gaye & Volkhan
Gaye & Yavuz
Gaye & Zeliha
Gaye & Beril
Hulya & idil
Hulya & Jale
Hulya & Lale
Hulya & Melis
Hulya & Selin
Hulya & Anil
idil & Jale
idil & Kadir
idil & Melis
idil & Onur
idil & Rasim
idil & Ulvi
idil & Yavuz
idil & Zeliha
idil & Anil
Jale & Kadir
Jale & Lale
Jale & Melis
Jale & Pelin
Jale & Rasim

Jale & Selin
Jale & Zeliha
Jale & Anil
Jale & Beril
Kadir & Lale
Kadir & Onur
Kadir & Pelin
Kadir & Rasim
Kadir & Volkan
Kadir & Yavuz
Lale & Melis
Lale & Pelin
Lale & Rasim
Lale & Selin
Lale & Tolga
Lale & Volkan
Lale & Yavuz
Lale & Zeliha
Lale & Anil
Lale & Beril
Melis & Onur
Melis & Pelin
Melis & Anil
Melis & Beril
Nedim & Onur
Nedim & Pelin
Nedim & Selin
Nedim & Tolga
Nedim & Ulvi
Nedim & Yavuz
Nedim & Anil
Nedim & Beril
Onur & Pelin
Onur & Tolga
Onur & Ulvi
Pelin & Selin
Pelin & Tolga
Pelin & Zeliha
Pelin & Beril
Rasim & Ulvi
Rasim & Anil
Tolga & Zeliha
Tolga & Beril
Ulvi & Volkan
Ulvi & Zeliha
Ulvi & Anil
Ulvi & Beril
Volkan & Anil
Yavuz & Zeliha
Yavuz & Anil
Zeliha & Anil
Anil & Beril

04/02/2021

TEZ_DOC

```
In [285]: for edge in sim3.edges:
    sim3.edges[edge]['frustrated'] = edge in cpu_imbalance
for node in sim3.nodes:
    sim3.nodes[node]['color'] = cpu_bicoloring[node]

print("3.PROBLEM (CPU):")
print('GRUP A: \n\t' + '\n\t'.join(list(person for (person, color) in cp
u_bicoloring.items() if (color == 0))))
print('GRUP B: \n\t' + '\n\t'.join(list(person for (person, color) in cp
u_bicoloring.items() if (color == 1))))
print('DENGESİZ/INSTABİL İLİŞKİLER: \n\t' + '\n\t'.join(list(x + " & " +
y for (x, y) in cpu_imbalance.keys())))
```

3.PROBLEM (CPU):
GRUP A:

Anıl
Ayşe
Barış
Eda
Jale
Kadir
Lale
Nedim
Pelin
Rasim
Selin
Tolga
Ulvi
Zeliha
İdil

GRUP B:

Beril
Ceren
Deniz
Fahrettin
Gaye
Hulya
Melis
Onur
Volkan
Yavuz

DENGESİZ/INSTABİL İLİŞKİLER:

Ayşe & Barış
Ayşe & Fahrettin
Ayşe & Hulya
Ayşe & Rasim
Ayşe & Selin
Ayşe & Tolga
Ayşe & Volkan
Ayşe & Zeliha
Ayşe & Beril
Barış & Ceren
Barış & Jale
Barış & Kadir
Barış & Selin
Barış & Yavuz
Barış & Beril
Ceren & İdil
Ceren & Jale
Ceren & Kadir
Ceren & Lale
Ceren & Melis
Ceren & Rasim
Ceren & Volkan
Ceren & Yavuz
Ceren & Zeliha
Deniz & Fahrettin
Deniz & Gaye
Deniz & Hulya
Deniz & İdil

04/02/2021

TEZ_DOC

Deniz & Melis
Deniz & Rasim
Deniz & Selin
Deniz & Ulvi
Deniz & Yavuz
Deniz & Anil
Eda & Fahrettin
Eda & Nedim
Eda & Onur
Eda & Selin
Eda & Tolga
Eda & Volkan
Eda & Yavuz
Fahrettin & Hulya
Fahrettin & Lale
Fahrettin & Melis
Fahrettin & Onur
Fahrettin & Pelin
Fahrettin & Rasim
Fahrettin & Selin
Fahrettin & Ulvi
Fahrettin & Volkan
Gaye & Hulya
Gaye & Onur
Gaye & Rasim
Gaye & Selin
Hulya & Melis
Hulya & Nedim
Hulya & Selin
İdil & Jale
İdil & Pelin
İdil & Selin
İdil & Tolga
İdil & Volkan
İdil & Anil
İdil & Beril
Jale & Lale
Jale & Onur
Jale & Tolga
Jale & Ulvi
Jale & Volkan
Jale & Yavuz
Jale & Anil
Kadir & Nedim
Kadir & Onur
Kadir & Pelin
Kadir & Rasim
Kadir & Volkan
Kadir & Yavuz
Kadir & Anil
Lale & Onur
Lale & Ulvi
Lale & Anil
Melis & Nedim
Melis & Onur
Melis & Pelin
Melis & Beril

localhost:8888/nbconvert/html/Macintosh HD/Users/egedursun/Desktop/Tez/TEZ_DOC.ipynb?download=false

80/146

04/02/2021

TEZ_DOC

Nedim & Rasim
 Nedim & Volkan
 Nedim & Zeliha
 Nedim & Anıl
 Onur & Pelin
 Onur & Tolga
 Onur & Ulvi
 Onur & Anıl
 Pelin & Selin
 Pelin & Tolga
 Pelin & Zeliha
 Pelin & Anıl
 Pelin & Beril
 Rasim & Ulvi
 Selin & Anıl
 Tolga & Zeliha
 Tolga & Anıl
 Tolga & Beril
 Ulvi & Volkan
 Ulvi & Zeliha
 Ulvi & Beril
 Yavuz & Zeliha
 Yavuz & Beril

D-Wave QPU/CPU Hibrit Çözümleyiciler ile Çeşitli Problemlerin Çözümlenmesi

Bu bölümde NetworkX kütüphanesi kullanılarak yaratılmış çizgeler ile ifade edilen problemleri, D-Wave'in bulut üzerinden sunduğu hibrit çözümleyici servislerini kullanarak çözümlemeye çalıştık. QPU/CPU hibrit çözümleyiciler Analog Kuantum Bilgisayarlarının ve Klasik CPU'ların işlem gücünü uygun noktalarda hibritleyerek daha büyük problem boyutlarında performansı artırma amacıyla gürültmektedir. Bu bölümde yukarıda ifade edilmiş problemlere kıyasla daha büyük boyutlu problemleri hibrit örneklendirici ile çözmeye çalışıp sonuçları raporladık.

```
In [287]: from dwave.system import LeapHybridSampler
hybrid_sampler = LeapHybridSampler()
```

Minimum Vertex Cover Problemi

Problem

04/02/2021

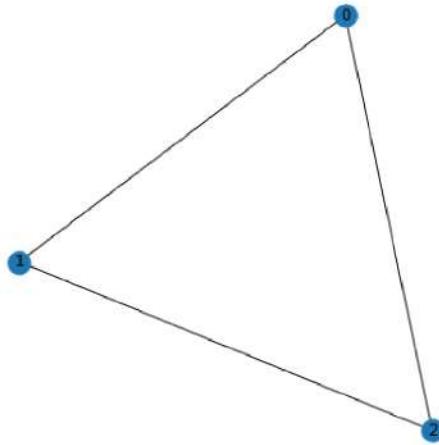
TEZ_DOC

```
In [301]: hvcl = nx.complete_graph(3)

for (u,v,w) in hvcl.edges(data=True):
    w['weight'] = random.randint(0,20)

plt.figure(figsize=(5,5))
nx.draw(hvcl, with_labels=True, font_weights='bold')

/Users/egedursun/opt/anaconda3/lib/python3.7/site-packages/networkx/drawing/nx_pylab.py:579: MatplotlibDeprecationWarning:
The iterable function was deprecated in Matplotlib 3.1 and will be removed in 3.3. Use np.iterable instead.
    if not cb.iterable(width):
```



QPU/CPU Hibrit ile Çözümleme

```
In [304]: c_start = time.time()

hybrid_result_a = dnx.min_vertex_cover(hvcl, hybrid_sampler)

c_end = time.time()

hybrid_result_time = (c_end - c_start)

print("1. PROBLEM - QPU/CPU HİBRİT ÇÖZÜM SÜRESİ: ", hybrid_result_time)
```

1. PROBLEM - QPU/CPU HİBRİT ÇÖZÜM SÜRESİ: 3676.800828933716

QPU ile Çözümleme

localhost:8888/nbconvert/html/Macintosh HD/Users/egedursun/Desktop/Tez/TEZ_DOC.ipynb?download=false

82/146

04/02/2021

TEZ_DOC

```
In [306]: c_start = time.time()
qpu_result_a = dnx.min_vertex_cover(hvcl, qpu_sampler)
c_end = time.time()
qpu_result_time = (c_end - c_start)
print("1. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)

1. PROBLEM - QPU ÇÖZÜM SÜRESİ: 4.501462697982788
```

CPU ile Çözümleme

```
In [307]: c_start = time.time()
cpu_result_a = dnx.min_vertex_cover(hvcl, cpu_sampler)
c_end = time.time()
cpu_result_time = (c_end - c_start)
print("1. PROBLEM - CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)

1. PROBLEM - CPU ÇÖZÜM SÜRESİ: 0.0014357566833496094
```

Çözümlerin Karşılaştırılması

```
In [312]: print("QPU/CPU HİBRİT SONUÇ: ", hybrid_result_a)
print("QPU SONUÇ: ", qpu_result_a)
print("CPU SONUÇ: ", cpu_result_a)

QPU/CPU HİBRİT SONUÇ: [0, 1]
QPU SONUÇ: [1, 2]
CPU SONUÇ: [1, 2]
```

Graph Coloring Problemı

Problem

04/02/2021

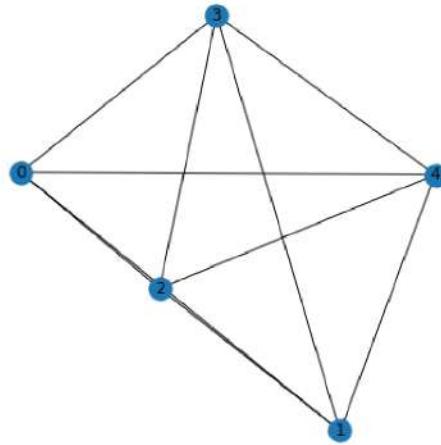
TEZ_DOC

```
In [331]: hymc = nx.complete_graph(5)

for (u,v,w) in hymc.edges(data=True):
    w['weight'] = random.randint(0,20)

plt.figure(figsize=(5,5))
nx.draw(hymc, with_labels=True, font_weights='bold')

/Users/egedursun/opt/anaconda3/lib/python3.7/site-packages/networkx/drawing/nx_pylab.py:579: MatplotlibDeprecationWarning:
The iterable function was deprecated in Matplotlib 3.1 and will be removed in 3.3. Use np.iterable instead.
    if not cb.iterable(width):
```



QPU/CPU Hibrit ile Çözümleme

```
In [332]: c_start = time.time()

hybrid_result_a = dnx.min_vertex_coloring(hymc, hybrid_sampler)

c_end = time.time()

hybrid_result_time = (c_end - c_start)

print("QPU/CPU HİBRİT ÇÖZÜM SÜRESİ: ", hybrid_result_time)
```

QPU/CPU HİBRİT ÇÖZÜM SÜRESİ: 14.24190092086792

QPU ile Çözümleme

localhost:8888/nbconvert/html/Macintosh HD/Users/egedursun/Desktop/Tez/TEZ_DOC.ipynb?download=false

84/146

04/02/2021

TEZ_DOC

```
In [333]: c_start = time.time()

qpu_result_a = dnx.min_vertex_coloring(hymc, qpu_sampler)

c_end = time.time()

qpu_result_time = (c_end - c_start)

print("QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)

QPU ÇÖZÜM SÜRESİ: 5.506086111068726
```

CPU ile Çözümleme

```
In [334]: c_start = time.time()

cpu_result_a = dnx.min_vertex_coloring(hymc, cpu_sampler)

c_end = time.time()

cpu_result_time = (c_end - c_start)

print("CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)

CPU ÇÖZÜM SÜRESİ: 123.29105687141418
```

Çözümlerin Karşılaştırılması

```
In [335]: print("QPU/CPU HİBRİT SONUÇ: ", hybrid_result_a)
print("QPU SONUÇ: ", qpu_result_a)
print("CPU SONUÇ: ", cpu_result_a)

QPU/CPU HİBRİT SONUÇ: {0: 4, 1: 0, 2: 1, 3: 3, 4: 2}
QPU SONUÇ: {0: 2, 1: 0, 2: 4, 4: 3}
CPU SONUÇ: {0: 3, 1: 4, 2: 1, 3: 2, 4: 0}
```

Maximum Clique Problemi

Problem

04/02/2021

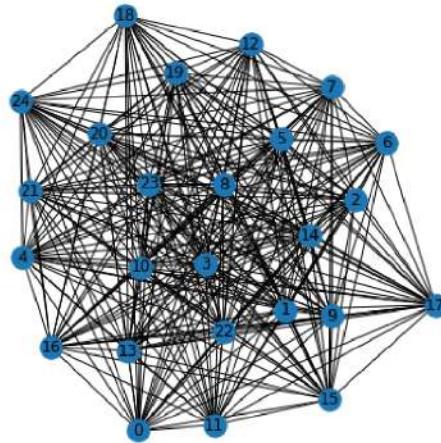
TEZ_DOC

```
In [345]: hymax = nx.complete_graph(25)

for (u,v,w) in hymax.edges(data=True):
    w['weight'] = random.randint(0,20)

plt.figure(figsize=(5,5))
nx.draw(hymax, with_labels=True, font_weights='bold')

/Users/egedursun/opt/anaconda3/lib/python3.7/site-packages/networkx/drawing/nx_pylab.py:579: MatplotlibDeprecationWarning:
The iterable function was deprecated in Matplotlib 3.1 and will be removed in 3.3. Use np.iterable instead.
    if not cb.iterable(width):
```



QPU/CPU Hibrit ile Çözümleme

```
In [346]: c_start = time.time()

hybrid_result_a = dnx.maximum_clique(hymax, hybrid_sampler)

c_end = time.time()

hybrid_result_time = (c_end - c_start)

print("QPU/CPU HİBRİT ÇÖZÜM SÜRESİ: ", hybrid_result_time)
```

QPU/CPU HİBRİT ÇÖZÜM SÜRESİ: 14.414310216903687

QPU ile Çözümleme

localhost:8888/nbconvert/html/Macintosh HD/Users/egedursun/Desktop/Tez/TEZ_DOC.ipynb?download=false

86/146

04/02/2021

TEZ_DOC

```
In [347]: c_start = time.time()

qpu_result_a = dnx.maximum_clique(hymax, qpu_sampler)

c_end = time.time()

qpu_result_time = (c_end - c_start)

print("GPU ÇÖZÜM SÜRESİ: ", qpu_result_time)

GPU ÇÖZÜM SÜRESİ: 3.5077292919158936
```

CPU ile Çözümleme

```
In [349]: c_start = time.time()

cpu_result_a = dnx.maximum_clique(hymax, cpu_sampler)

c_end = time.time()

cpu_result_time = (c_end - c_start)

print("CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)

CPU ÇÖZÜM SÜRESİ: 65.40786409378052
```

Çözümlerin Karşılaştırılması

```
In [350]: print("GPU/CPU HİBRİT SONUÇ: ", hybrid_result_a)
print("GPU SONUÇ: ", qpu_result_a)
print("CPU SONUÇ: ", cpu_result_a)

GPU/CPU HİBRİT SONUÇ: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24]
GPU SONUÇ: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24]
CPU SONUÇ: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24]
```

Minimum Maximal Matching Problemi

Problem

04/02/2021

TEZ_DOC

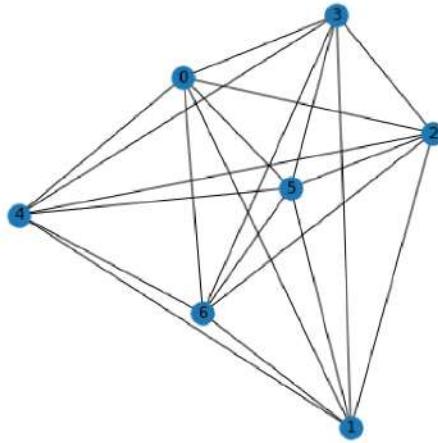
```
In [355]: hymin = nx.complete_graph(7)

for (u,v,w) in hymin.edges(data=True):
    w['weight'] = random.randint(0,20)

plt.figure(figsize=(5,5))
nx.draw(hymin, with_labels=True, font_weights='bold')

/Users/egedursun/opt/anaconda3/lib/python3.7/site-packages/networkx/drawing/nx_pylab.py:579: MatplotlibDeprecationWarning:
The iterable function was deprecated in Matplotlib 3.1 and will be removed in 3.3. Use np.iterable instead.

if not cb.iterable(width):
```



QPU/CPU Hibrit ile Çözümleme

```
In [356]: c_start = time.time()

hybrid_result_a = dnx.min_maximal_matching(hymin, hybrid_sampler)

c_end = time.time()

hybrid_result_time = (c_end - c_start)

print("QPU/CPU HİBRİT ÇÖZÜM SÜRESİ: ", hybrid_result_time)
```

QPU/CPU HİBRİT ÇÖZÜM SÜRESİ: 14.226783990859985

QPU ile Çözümleme

localhost:8888/nbconvert/html/Macintosh HD/Users/egedursun/Desktop/Tez/TEZ_DOC.ipynb?download=false

88/146

04/02/2021

TEZ_DOC

```
In [357]: c_start = time.time()

qpu_result_a = dnx.min_maximal_matching(hymin, qpu_sampler)

c_end = time.time()

qpu_result_time = (c_end - c_start)

print("QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)

QPU ÇÖZÜM SÜRESİ: 5.954056978225708
```

CPU ile Çözümleme

```
In [358]: c_start = time.time()

cpu_result_a = dnx.min_maximal_matching(hymin, cpu_sampler)

c_end = time.time()

cpu_result_time = (c_end - c_start)

print("CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)

CPU ÇÖZÜM SÜRESİ: 7.748928070068359
```

Çözümlerin Karşılaştırılması

```
In [359]: print("QPU/+"
         "-CPU HİBRİT SONUÇ: ", hybrid_result_a)
print("QPU SONUÇ: ", qpu_result_a)
print("CPU SONUÇ: ", cpu_result_a)

QPU/CPU HİBRİT SONUÇ: {(4, 5), (2, 6), (1, 3)}
QPU SONUÇ: {(1, 2), (4, 5), (1, 6), (2, 3), (0, 4), (2, 5), (3, 4)}
CPU SONUÇ: {(1, 5), (4, 6), (0, 2)}
```

Maximum Cut Problemi

Problem

04/02/2021

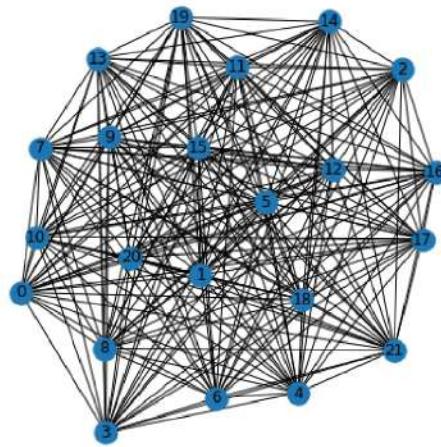
TEZ_DOC

```
In [376]: hymcut = nx.complete_graph(22)

for (u,v,w) in hymcut.edges(data=True):
    w['weight'] = random.randint(0,20)

plt.figure(figsize=(5,5))
nx.draw(hymcut, with_labels=True, font_weights='bold')

/Users/egedursun/opt/anaconda3/lib/python3.7/site-packages/networkx/drawing/nx_pylab.py:579: MatplotlibDeprecationWarning:
The iterable function was deprecated in Matplotlib 3.1 and will be removed in 3.3. Use np.iterable instead.
    if not cb.iterable(width):
```



QPU/CPU Hibrit ile Çözümleme

```
In [377]: c_start = time.time()

hybrid_result_a = dnx.maximum_cut(hymcut, hybrid_sampler)

c_end = time.time()

hybrid_result_time = (c_end - c_start)

print("QPU/CPU HİBRİT ÇÖZÜM SÜRESİ: ", hybrid_result_time)
```

QPU/CPU HİBRİT ÇÖZÜM SÜRESİ: 15.238375902175903

QPU ile Çözümleme

localhost:8888/nbconvert/html/Macintosh HD/Users/egedursun/Desktop/Tez/TEZ_DOC.ipynb?download=false

90/146

04/02/2021

TEZ_DOC

```
In [378]: c_start = time.time()

qpu_result_a = dnx.maximum_cut(hymcut, qpu_sampler)

c_end = time.time()

qpu_result_time = (c_end - c_start)

print("QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)

QPU ÇÖZÜM SÜRESİ: 8.182854175567627
```

CPU ile Çözümleme

```
In [379]: c_start = time.time()

cpu_result_a = dnx.maximum_cut(hymcut, cpu_sampler)

c_end = time.time()

cpu_result_time = (c_end - c_start)

print("CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)

CPU ÇÖZÜM SÜRESİ: 27.116833925247192
```

Çözümlerin Karşılaştırılması

```
In [380]: print("QPU/CPU HİBRİT SONUÇ: ", hybrid_result_a)
print("QPU SONUÇ: ", qpu_result_a)
print("CPU SONUÇ: ", cpu_result_a)

QPU/CPU HİBRİT SONUÇ: {4, 5, 7, 8, 9, 10, 13, 14, 15, 16, 21}
QPU SONUÇ: {5, 7, 9, 11, 12, 13, 15, 17, 18, 20, 21}
CPU SONUÇ: {1, 2, 5, 6, 8, 10, 11, 12, 16, 17, 20}
```

Traveling Salesperson Problemi

Problem

04/02/2021

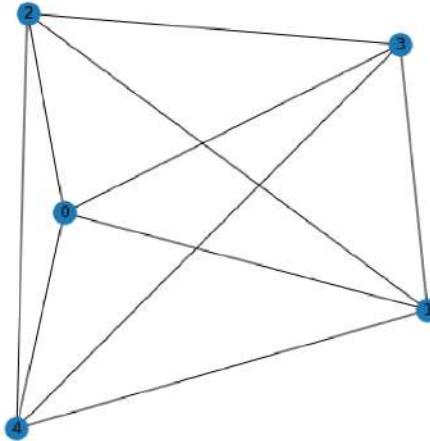
TEZ_DOC

```
In [384]: hysp = nx.complete_graph(5)

for (u,v,w) in hysp.edges(data=True):
    w['weight'] = random.randint(0,20)

plt.figure(figsize=(5,5))
nx.draw(hysp, with_labels=True, font_weights='bold')

/Users/egedursun/opt/anaconda3/lib/python3.7/site-packages/networkx/drawing/nx_pylab.py:579: MatplotlibDeprecationWarning:
The iterable function was deprecated in Matplotlib 3.1 and will be removed in 3.3. Use np.iterable instead.
    if not cb.iterable(width):
```



QPU/CPU Hibrit ile Çözümleme

```
In [383]: c_start = time.time()

hybrid_result_a = dnx.traveling_salesperson(hysp, hybrid_sampler)

c_end = time.time()

hybrid_result_time = (c_end - c_start)

print("QPU/CPU HİBRİT ÇÖZÜM SÜRESİ: ", hybrid_result_time)
```

QPU/CPU HİBRİT ÇÖZÜM SÜRESİ: 16.50662088394165

QPU ile Çözümleme

localhost:8888/nbconvert/html/Macintosh HD/Users/egedursun/Desktop/Tez/TEZ_DOC.ipynb?download=false

92/146

04/02/2021

TEZ_DOC

```
In [385]: c_start = time.time()
qpu_result_a = dnx.traveling_salesperson(hysp, qpu_sampler)
c_end = time.time()
qpu_result_time = (c_end - c_start)
print("QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)

QPU ÇÖZÜM SÜRESİ:  9.78623914718628
```

CPU ile Çözümleme

```
In [386]: c_start = time.time()
cpu_result_a = dnx.traveling_salesperson(hysp, cpu_sampler)
c_end = time.time()
cpu_result_time = (c_end - c_start)
print("CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)

CPU ÇÖZÜM SÜRESİ:  185.0526249408722
```

Çözümlerin Karşılaştırılması

```
In [387]: print("QPU/CPU HİBRİT SONUÇ: ", hybrid_result_a)
print("QPU SONUÇ: ", qpu_result_a)
print("CPU SONUÇ: ", cpu_result_a)

QPU/CPU HİBRİT SONUÇ: [2, 9, 5, 7, 6, 0, 1, 3, 8, 4]
QPU SONUÇ: [3, 2, 3, 1, 4]
CPU SONUÇ: [2, 1, 0, 3, 4]
```

Structural Imbalance Problemı

Problem

04/02/2021

TEZ_DOC

```
In [395]: hysim = nx.complete_graph(20)
hysim.add_edges_from([(u, v, {'sign': 2*random.randint(0, 1) - 1}) for u
, v in hysim.edges])

nx.relabel_nodes(hysim,
                 {0: 'Ayse', 1: 'Baris', 2: 'Ceren', 3: 'Deniz', 4: 'Eda'
, 5: 'Fahrettin', 6: 'Gaye',
                 7: 'Halil', 8: 'Idil', 9: 'Jale', 10: 'Kadir', 11: 'Lale', 1
2: 'Melis', 13: 'Nedim',
                 14: 'Onur', 15: 'Pelin', 16: 'Rasim', 17: 'Selin', 18: 'Teki
n', 19: 'Ulvi'},
                 copy=False)

print('Pozitif (Arkadaşcil) ilişkiler: \n\t' + '\n\t'.join(
    list(x + " & " + y for (x, y, sign) in hysim.edges(data='sign') if (
sign == 1))))
print('Negatif (Düşmancıl) ilişkiler: \n\t' + '\n\t'.join(
    list(x + " & " + y for (x, y, sign) in hysim.edges(data='sign') if (
sign == -1))))
```

Pozitif (Arkadaşcil) ilişkiler:
Ayse & Baris
Ayse & Deniz
Ayse & Halil
Ayse & Jale
Ayse & Kadir
Ayse & Lale
Ayse & Melis
Ayse & Onur
Ayse & Selin
Baris & Gaye
Baris & Jale
Baris & Kadir
Baris & Nedim
Baris & Pelin
Baris & Tekin
Baris & Ulvi
Ceren & Deniz
Ceren & Fahrettin
Ceren & Halil
Ceren & Idil
Ceren & Jale
Ceren & Kadir
Ceren & Nedim
Ceren & Onur
Ceren & Pelin
Ceren & Selin
Ceren & Tekin
Ceren & Ulvi
Deniz & Eda
Deniz & Gaye
Deniz & Halil
Deniz & Jale
Deniz & Lale
Deniz & Melis
Deniz & Nedim
Deniz & Pelin
Deniz & Tekin
Deniz & Ulvi
Eda & Fahrettin
Eda & Halil
Eda & Lale
Eda & Melis
Eda & Onur
Eda & Pelin
Eda & Ulvi
Fahrettin & Halil
Fahrettin & Jale
Fahrettin & Melis
Fahrettin & Nedim
Fahrettin & Rasim
Fahrettin & Selin
Fahrettin & Ulvi
Gaye & Melis
Gaye & Nedim
Gaye & Selin
Halil & Idil

Halil & Lale
Halil & Melis
Halil & Pelin
Halil & Rasim
Idil & Kadir
Idil & Lale
Idil & Nedim
Idil & Tekin
Idil & Ulvi
Jale & Melis
Jale & Nedim
Jale & Onur
Jale & Selin
Jale & Tekin
Jale & Ulvi
Kadir & Lale
Kadir & Pelin
Kadir & Selin
Lale & Onur
Lale & Rasim
Lale & Tekin
Melis & Rasim
Melis & Selin
Melis & Tekin
Onur & Pelin
Onur & Rasim
Onur & Selin
Onur & Ulvi
Pelin & Rasim
Pelin & Selin
Pelin & Tekin
Rasim & Selin
Rasim & Ulvi
Selin & Tekin

Negatif (Düşmancıl) ilişkiler:

Ayse & Ceren
Ayse & Eda
Ayse & Fahrettin
Ayse & Gaye
Ayse & Idil
Ayse & Nedim
Ayse & Pelin
Ayse & Rasim
Ayse & Tekin
Ayse & Ulvi
Baris & Ceren
Baris & Deniz
Baris & Eda
Baris & Fahrettin
Baris & Halil
Baris & Idil
Baris & Lale
Baris & Melis
Baris & Onur
Baris & Rasim
Baris & Selin
Ceren & Eda

Ceren & Gaye
Ceren & Lale
Ceren & Melis
Ceren & Rasim
Deniz & Fahrettin
Deniz & Idil
Deniz & Kadir
Deniz & Onur
Deniz & Rasim
Deniz & Selin
Eda & Gaye
Eda & Idil
Eda & Jale
Eda & Kadir
Eda & Nedim
Eda & Rasim
Eda & Selin
Eda & Tekin
Fahrettin & Gaye
Fahrettin & Idil
Fahrettin & Kadir
Fahrettin & Lale
Fahrettin & Onur
Fahrettin & Pelin
Fahrettin & Tekin
Gaye & Halil
Gaye & Idil
Gaye & Jale
Gaye & Kadir
Gaye & Lale
Gaye & Onur
Gaye & Pelin
Gaye & Rasim
Gaye & Tekin
Gaye & Ulvi
Halil & Jale
Halil & Kadir
Halil & Nedim
Halil & Onur
Halil & Selin
Halil & Tekin
Halil & Ulvi
Idil & Jale
Idil & Melis
Idil & Onur
Idil & Pelin
Idil & Rasim
Idil & Selin
Jale & Kadir
Jale & Lale
Jale & Pelin
Jale & Rasim
Kadir & Melis
Kadir & Nedim
Kadir & Onur
Kadir & Rasim
Kadir & Tekin

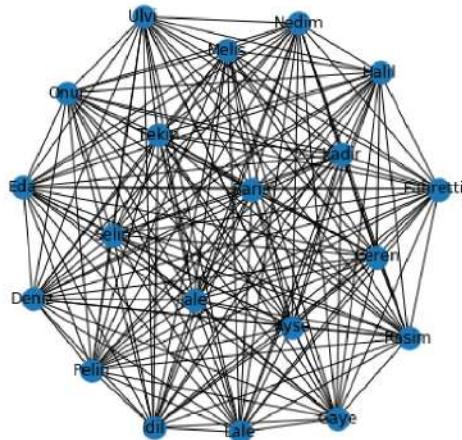
04/02/2021

TEZ_DOC

```
Kadir & Ulvi
Lale & Melis
Lale & Nedim
Lale & Pelin
Lale & Selin
Lale & Ulvi
Melis & Nedim
Melis & Onur
Melis & Pelin
Melis & Ulvi
Nedim & Onur
Nedim & Pelin
Nedim & Rasim
Nedim & Selin
Nedim & Tekin
Nedim & Ulvi
Onur & Tekin
Pelin & Ulvi
Rasim & Tekin
Selin & Ulvi
Tekin & Ulvi
```

```
In [396]: plt.figure(figsize=(5,5))
nx.draw(hysim, with_labels=True, font_weights='bold')

/Users/egedursun/opt/anaconda3/lib/python3.7/site-packages/networkx/drawing/nx_pylab.py:579: MatplotlibDeprecationWarning:
The iterable function was deprecated in Matplotlib 3.1 and will be removed in 3.3. Use np.iterable instead.
    if not cb.iterable(width):
```



QPU/CPU Hibrit ile Çözümleme

localhost:8888/nbconvert/html/Macintosh HD/Users/egedursun/Desktop/Tez/TEZ_DOC.ipynb?download=false

98/146

04/02/2021

TEZ_DOC

```
In [397]: c_start = time.time()
hybrid_imbalance_a, hybrid_bicoloring_a = dnx.structural_imbalance(hysim,
, hybrid_sampler)
c_end = time.time()
hybrid_result_time = (c_end - c_start)
print("QPU/CPU HİBRİT ÇÖZÜM SÜRESİ: ", hybrid_result_time)
```

QPU/CPU HİBRİT ÇÖZÜM SÜRESİ: 14.653132200241089

QPU ile Çözümleme

```
In [398]: c_start = time.time()
qpu_imbalance_a, qpu_bicoloring_a = dnx.structural_imbalance(hysim, qpu_
sampler)
c_end = time.time()
qpu_result_time = (c_end - c_start)
print("QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)
```

QPU ÇÖZÜM SÜRESİ: 5.697700262069702

CPU ile Çözümleme

```
In [399]: c_start = time.time()
cpu_imbalance_a, cpu_bicoloring_a = dnx.structural_imbalance(hysim, cpu_
sampler)
c_end = time.time()
cpu_result_time = (c_end - c_start)
print("CPU ÇÖZÜM SÜRESİ: ", cpu_result_time)
```

CPU ÇÖZÜM SÜRESİ: 3.2858550548553467

Çözümlerin Karşılaştırılması

04/02/2021

TEZ_DOC

```
In [401]: for edge in hysim.edges:
    hysim.edges[edge]['frustrated'] = edge in hybrid_imbalance_a
for node in hysim.nodes:
    hysim.nodes[node]['color'] = hybrid_bicoloring_a[node]

print("PROBLEM (QPU/CPU):")
print('GRUP A: \n\t' + '\n\t'.join(
    list(person for (person, color) in hybrid_bicoloring_a.items() if (color == 0))))
print('GRUP B: \n\t' + '\n\t'.join(
    list(person for (person, color) in hybrid_bicoloring_a.items() if (color == 1))))
print('DENGESİZ/INSTABİL İLİŞKİLER: \n\t' + '\n\t'.join(
    list(x + " & " + y for (x, y) in hybrid_imbalance_a.keys())))
```

04/02/2021

TEZ_DOC

PROBLEM (QPU/CPU):**GRUP A:**

Baris
Gaye
Idil
Kadir
Lale
Nedim
Tekin

GRUP B:

Ayse
Ceren
Deniz
Eda
Fahrettin
Halil
Jale
Melis
Onur
Pelin
Rasim
Selin
Ulvi

DENGESİZ/INSTABİL İLİŞKİLER:

Ayse & Baris
Ayse & Ceren
Ayse & Eda
Ayse & Fahrettin
Ayse & Kadir
Ayse & Lale
Ayse & Pelin
Ayse & Rasim
Ayse & Ulvi
Baris & Idil
Baris & Jale
Baris & Lale
Baris & Pelin
Baris & Ulvi
Ceren & Eda
Ceren & Idil
Ceren & Kadir
Ceren & Melis
Ceren & Nedim
Ceren & Rasim
Ceren & Tekin
Deniz & Fahrettin
Deniz & Gaye
Deniz & Lale
Deniz & Nedim
Deniz & Onur
Deniz & Rasim
Deniz & Selin
Deniz & Tekin
Eda & Jale
Eda & Lale
Eda & Rasim
Eda & Selin

04/02/2021

TEZ_DOC

Fahrettin & Nedim
Fahrettin & Onur
Fahrettin & Pelin
Gaye & Idil
Gaye & Kadir
Gaye & Lale
Gaye & Melis
Gaye & Selin
Gaye & Tekin
Halil & Idil
Halil & Jale
Halil & Lale
Halil & Onur
Halil & Selin
Halil & Ulvi
Idil & Ulvi
Jale & Nedim
Jale & Pelin
Jale & Rasim
Jale & Tekin
Kadir & Nedim
Kadir & Pelin
Kadir & Selin
Kadir & Tekin
Lale & Nedim
Lale & Onur
Lale & Rasim
Melis & Onur
Melis & Pelin
Melis & Tekin
Melis & Ulvi
Nedim & Tekin
Pelin & Tekin
Pelin & Ulvi
Selin & Tekin
Selin & Ulvi

04/02/2021

TEZ_DOC

```
In [403]: for edge in hysim.edges:
    hysim.edges[edge]['frustrated'] = edge in qpu_imbalance_a
for node in hysim.nodes:
    hysim.nodes[node]['color'] = qpu_bicoloring_a[node]

print("PROBLEM (QPU):")
print('GRUP A: \n\t' + '\n\t'.join(list(person for (person, color) in qp_u_bicoloring_a.items() if (color == 0))))
print('GRUP B: \n\t' + '\n\t'.join(list(person for (person, color) in qp_u_bicoloring_a.items() if (color == 1))))
print('DENGESİZ/INSTABİL İLİŞKİLER: \n\t' + '\n\t'.join(list(x + " & " +
y for (x, y) in qpu_imbalance_a.keys()))))
```

04/02/2021

TEZ_DOC

PROBLEM (QPU):
GRUP A:

Eda
Halil
Idil
Kadir
Lale
Onur
Pelin
Rasim

GRUP B:

Ayse
Baris
Ceren
Deniz
Fahrettin
Gaye
Jale
Melis
Nedim
Selin
Tekin
Ulvi

DENGESİZ/INSTABİL İLİŞKİLER:

Ayse & Ceren
Ayse & Fahrettin
Ayse & Gaye
Ayse & Halil
Ayse & Kadir
Ayse & Lale
Ayse & Nedim
Ayse & Onur
Ayse & Tekin
Ayse & Ulvi
Baris & Ceren
Baris & Deniz
Baris & Fahrettin
Baris & Kadir
Baris & Melis
Baris & Pelin
Baris & Selin
Ceren & Gaye
Ceren & Halil
Ceren & Idil
Ceren & Kadir
Ceren & Melis
Ceren & Onur
Ceren & Pelin
Deniz & Eda
Deniz & Fahrettin
Deniz & Halil
Deniz & Lale
Deniz & Pelin
Deniz & Selin
Eda & Fahrettin
Eda & Idil
Eda & Kadir

localhost:8888/nbconvert/html/Macintosh HD/Users/egedursun/Desktop/Tez/TEZ_DOC.ipynb?download=false

104/146

04/02/2021

TEZ_DOC

Eda & Melis
Eda & Rasim
Eda & Ulvi
Fahrettin & Gaye
Fahrettin & Halil
Fahrettin & Rasim
Fahrettin & Tekin
Gaye & Jale
Gaye & Tekin
Gaye & Ulvi
Halil & Kadir
Halil & Melis
Halil & Onur
Idil & Nedim
Idil & Onur
Idil & Pelin
Idil & Rasim
Idil & Tekin
Idil & Ulvi
Jale & Onur
Kadir & Onur
Kadir & Rasim
Kadir & Selin
Lale & Pelin
Lale & Tekin
Melis & Nedim
Melis & Rasim
Melis & Ulvi
Nedim & Selin
Nedim & Tekin
Nedim & Ulvi
Onur & Selin
Onur & Ulvi
Pelin & Selin
Pelin & Tekin
Rasim & Selin
Rasim & Ulvi
Selin & Ulvi
Tekin & Ulvi

04/02/2021

TEZ_DOC

```
In [404]: for edge in hysim.edges:
    hysim.edges[edge]['frustrated'] = edge in cpu_imbalance_a
for node in hysim.nodes:
    hysim.nodes[node]['color'] = cpu_bicoloring_a[node]

print("PROBLEM (CPU):")
print('GRUP A: \n\t' + '\n\t'.join(list(person for (person, color) in cp
u_bicoloring_a.items() if (color == 0))))
print('GRUP B: \n\t' + '\n\t'.join(list(person for (person, color) in cp
u_bicoloring_a.items() if (color == 1))))
print('DENGESİZ/INSTABİL İLİŞKİLER: \n\t' + '\n\t'.join(list(x + " & "
y for (x, y) in cpu_imbalance_a.keys())))
```

04/02/2021

TEZ_DOC

PROBLEM (CPU):
GRUP A:

Ayse
Baris
Gaye
Idil
Kadir
Lale
Nedim

GRUP B:

Ceren
Deniz
Eda
Fahrettin
Halil
Jale
Melis
Onur
Pelin
Rasim
Selin
Tekin
Ulvi

DENGESİZ/INSTABİL İLİŞKİLER:

Ayse & Deniz
Ayse & Gaye
Ayse & Halil
Ayse & Idil
Ayse & Jale
Ayse & Melis
Ayse & Nedim
Ayse & Onur
Ayse & Selin
Baris & Idil
Baris & Jale
Baris & Lale
Baris & Pelin
Baris & Tekin
Baris & Ulvi
Ceren & Eda
Ceren & Idil
Ceren & Kadir
Ceren & Melis
Ceren & Nedim
Ceren & Rasim
Deniz & Fahrettin
Deniz & Gaye
Deniz & Lale
Deniz & Nedim
Deniz & Onur
Deniz & Rasim
Deniz & Selin
Eda & Jale
Eda & Lale
Eda & Rasim
Eda & Selin
Eda & Tekin

localhost:8888/nbconvert/html/Macintosh HD/Users/egedursun/Desktop/Tez/TEZ_DOC.ipynb?download=false

107/146

04/02/2021

TEZ_DOC

```
Fahrettin & Nedim
Fahrettin & Onur
Fahrettin & Pelin
Fahrettin & Tekin
Gaye & Idil
Gaye & Kadir
Gaye & Lale
Gaye & Melis
Gaye & Selin
Halil & Idil
Halil & Jale
Halil & Lale
Halil & Onur
Halil & Selin
Halil & Tekin
Halil & Ulvi
Idil & Tekin
Idil & Ulvi
Jale & Nedim
Jale & Pelin
Jale & Rasim
Kadir & Nedim
Kadir & Pelin
Kadir & Selin
Lale & Nedim
Lale & Onur
Lale & Rasim
Lale & Tekin
Melis & Onur
Melis & Pelin
Melis & Ulvi
Onur & Tekin
Pelin & Ulvi
Rasim & Tekin
Selin & Ulvi
Tekin & Ulvi
```

Yapısal Dengesizlik Problemlerinin Çözümlenmesi

Bu bölümde NetworkX kütüphanesi kullanılarak yaratılan çeşitli çizgeler üzerinden, büyük boyutlu ve gerçek dünyada karşılaşma olanağı olan yapısal dengesizlik problemlerine D-Wave QPU/CPU Hibriti ve QPU kullanarak çözümler getirmeye çalışacak ve performanslarını kaydedeceğiz.

```
In [415]: import string
```

1. Problem

04/02/2021

TEZ_DOC

```
In [582]: sil = nx.complete_graph(35)
sil.add_edges_from([(u, v, {'sign': 2*random.randint(0, 1) - 1}) for u,
v in sil.edges])

people = {}

for i in range(0, 35):

    people[i] = ''.join(random.choice(string.ascii_uppercase) for _ in range(8))

nx.relabel_nodes(sil,
                 people,
                 copy=False)

#print('Positif (Arkadaşcil) ilişkiler: \n\t' + '\n\t'.join(list(x + " &
" + y for (x, y, sign) in sil.edges(data='sign') if (sign == 1))))
#print('Negatif (Düşmancıl) ilişkiler: \n\t' + '\n\t'.join(list(x + " &
" + y for (x, y, sign) in sil.edges(data='sign') if (sign == -1))))
```

Out[582]: <networkx.classes.graph.Graph at 0x1b8a48c90>

04/02/2021

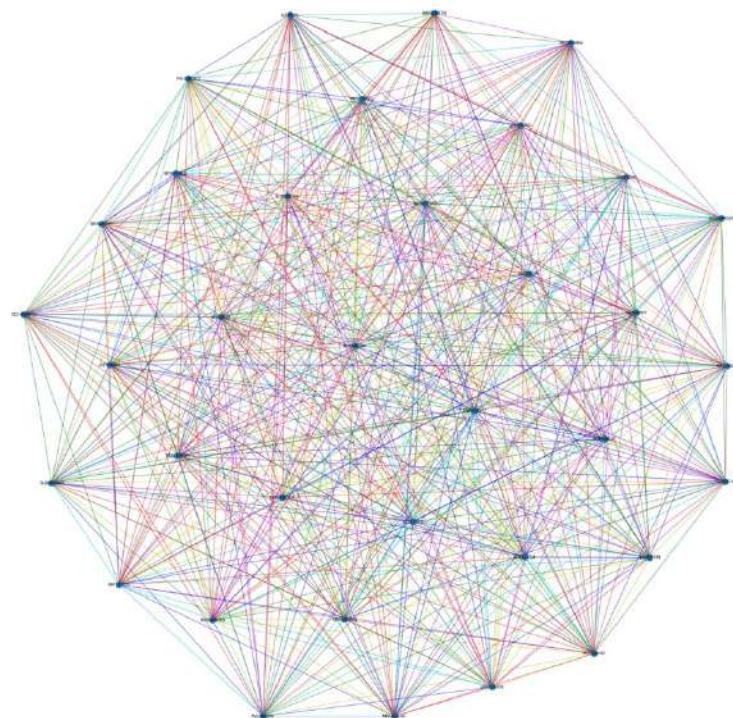
TEZ_DOC

```
In [583]: plt.figure(figsize=(30,30))

l = ["r", "g", "b", "y", "c", "m"]

colors = []
for i in range(0, len(sil.edges)):
    colors.append(random.choice(l))
colors

nx.draw(sil, with_labels=True, edge_color=tuple(colors), font_weights='bold')
```



2. Problem

localhost:8888/nbconvert/html/Macintosh HD/Users/egedursun/Desktop/Tez/TEZ_DOC.ipynb?download=false

110/146

04/02/2021

TEZ_DOC

```
In [578]: si2 = nx.complete_graph(20)
si2.add_edges_from([(u, v, {'sign': 2*random.randint(0, 1) - 1}) for u,
v in si2.edges])

people = {}

for i in range(0, 20):

    people[i] = ''.join(random.choice(string.ascii_uppercase) for _ in range(8))

nx.relabel_nodes(si2,
                 people,
                 copy=False)

#print('Positif (Arkadaşcil) ilişkiler: \n\t' + '\n\t'.join(list(x + " &
" + y for (x, y, sign) in sil.edges(data='sign') if (sign == 1))))
#print('Negatif (Düşmancıl) ilişkiler: \n\t' + '\n\t'.join(list(x + " &
" + y for (x, y, sign) in sil.edges(data='sign') if (sign == -1))))
```

Out[578]: <networkx.classes.graph.Graph at 0x1d59e6350>

04/02/2021

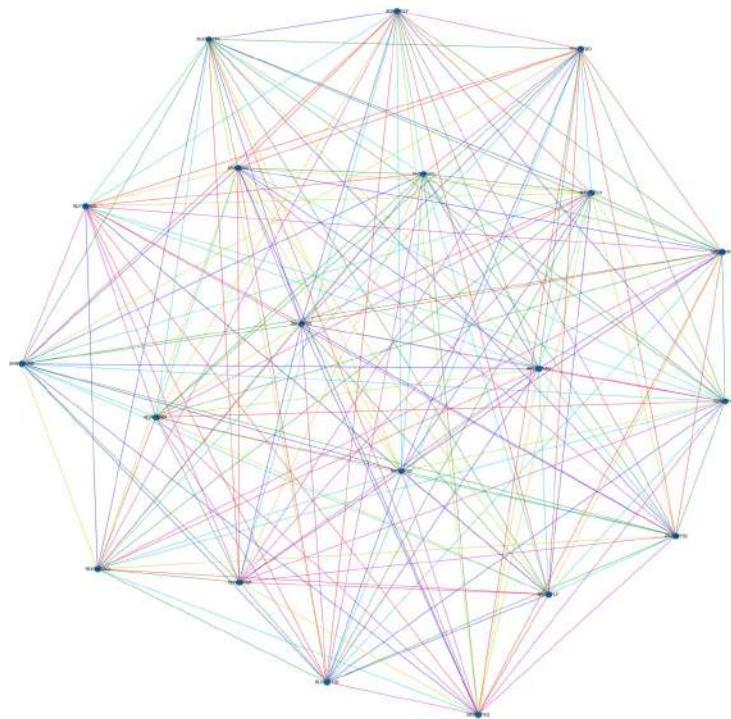
TEZ_DOC

```
In [579]: plt.figure(figsize=(30,30))

l = ["r", "g", "b", "y", "c", "m"]

colors = []
for i in range(0, len(si2.edges)):
    colors.append(random.choice(l))
colors

nx.draw(si2, with_labels=True, edge_color=tuple(colors), font_weights='bold')
```



3. Problem

localhost:8888/nbconvert/html/Macintosh HD/Users/egedursun/Desktop/Tez/TEZ_DOC.ipynb?download=false

112/146

04/02/2021

TEZ_DOC

```
In [580]: si3 = nx.complete_graph(30)
si3.add_edges_from([(u, v, {'sign': 2*random.randint(0, 1) - 1}) for u,
v in si3.edges])

people = {}

for i in range(0, 30):

    people[i] = ''.join(random.choice(string.ascii_uppercase) for _ in range(8))

nx.relabel_nodes(si3,
                 people,
                 copy=False)

#print('Positif (Arkadaşcil) İlişkiler: \n\t' + '\n\t'.join(list(x + " &
" + y for (x, y, sign) in sil.edges(data='sign') if (sign == 1))))
#print('Negatif (Düşmancıl) İlişkiler: \n\t' + '\n\t'.join(list(x + " &
" + y for (x, y, sign) in sil.edges(data='sign') if (sign == -1))))
```

Out[580]: <networkx.classes.graph.Graph at 0x1d59d6890>

04/02/2021

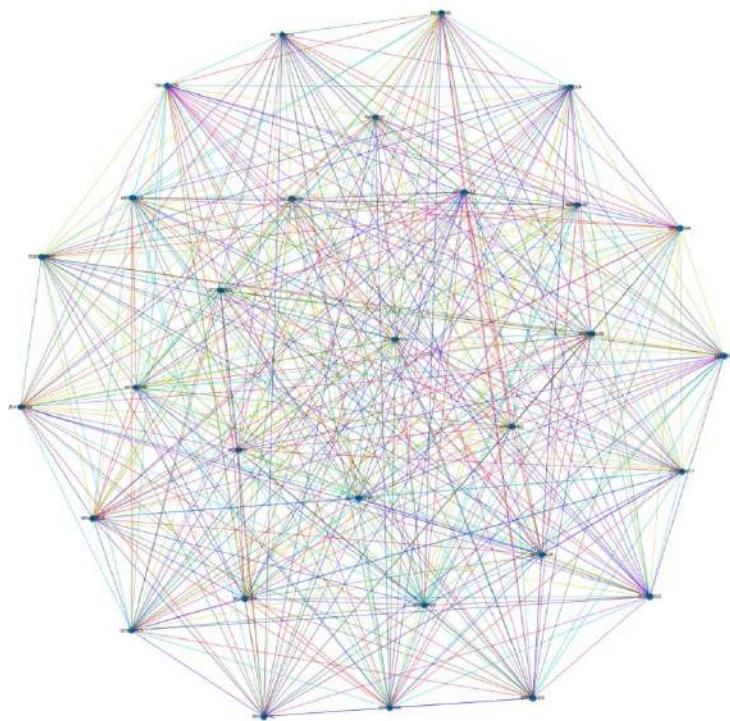
TEZ_DOC

```
In [581]: plt.figure(figsize=(30,30))

l = ["r", "g", "b", "y", "c", "m"]

colors = []
for i in range(0, len(si3.edges)):
    colors.append(random.choice(l))
colors

nx.draw(si3, with_labels=True, edge_color=tuple(colors), font_weights='bold')
```



QPU ile Çözümleme

Problem 1

localhost:8888/nbconvert/html/Macintosh HD/Users/egedursun/Desktop/Tez/TEZ_DOC.ipynb?download=false

114/146

04/02/2021

TEZ_DOC

```
In [584]: c_start = time.time()

qpu_imbalance_a, qpu_bicoloring_a = dnx.structural_imbalance(si1, qpu_sampler)

c_end = time.time()

qpu_result_time = (c_end - c_start)

print("1. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)
```

1. PROBLEM - QPU ÇÖZÜM SÜRESİ: 11.170470237731934

Problem 2

```
In [587]: c_start = time.time()

qpu_imbalance_b, qpu_bicoloring_b = dnx.structural_imbalance(si2, qpu_sampler)

c_end = time.time()

qpu_result_time = (c_end - c_start)

print("2. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)
```

2. PROBLEM - QPU ÇÖZÜM SÜRESİ: 5.7992589473724365

Problem 3

```
In [588]: c_start = time.time()

qpu_imbalance_c, qpu_bicoloring_c = dnx.structural_imbalance(si3, qpu_sampler)

c_end = time.time()

qpu_result_time = (c_end - c_start)

print("3. PROBLEM - QPU ÇÖZÜM SÜRESİ: ", qpu_result_time)
```

3. PROBLEM - QPU ÇÖZÜM SÜRESİ: 7.823553085327148

QPU/CPU Hibrit ile Çözümleme**Problem 1**

localhost:8888/nbconvert/html/Macintosh HD/Users/egedursun/Desktop/Tez/TEZ_DOC.ipynb?download=false

115/146

04/02/2021

TEZ_DOC

```
In [589]: c_start = time.time()

hybrid_imbalance_a, hybrid_bicoloring_a = dnx.structural_imbalance(si1,
hybrid_sampler)

c_end = time.time()

hybrid_result_time = (c_end - c_start)

print("1. PROBLEM - QPU/CPU HİBRİT ÇÖZÜM SÜRESİ: ", hybrid_result_time)
```

1. PROBLEM - QPU/CPU HİBRİT ÇÖZÜM SÜRESİ: 11.206799030303955

Problem 2

```
In [590]: c_start = time.time()

hybrid_imbalance_b, hybrid_bicoloring_b = dnx.structural_imbalance(si2,
hybrid_sampler)

c_end = time.time()

hybrid_result_time = (c_end - c_start)

print("2. PROBLEM - QPU/CPU HİBRİT ÇÖZÜM SÜRESİ: ", hybrid_result_time)
```

2. PROBLEM - QPU/CPU HİBRİT ÇÖZÜM SÜRESİ: 15.674062252044678

Problem 3

```
In [591]: c_start = time.time()

hybrid_imbalance_c, hybrid_bicoloring_c = dnx.structural_imbalance(si3,
hybrid_sampler)

c_end = time.time()

hybrid_result_time = (c_end - c_start)

print("3. PROBLEM - QPU/CPU HİBRİT ÇÖZÜM SÜRESİ: ", hybrid_result_time)
```

3. PROBLEM - QPU/CPU HİBRİT ÇÖZÜM SÜRESİ: 16.554741144180298

Çözümlerin Karşılaştırılması**1. Problem**

localhost:8888/nbconvert/html/Macintosh HD/Users/egedursun/Desktop/Tez/TEZ_DOC.ipynb?download=false

116/146

04/02/2021

TEZ_DOC

```
In [592]: for edge in sil.edges:
    sil.edges[edge]['frustrated'] = edge in qpu_imbalance_a
for node in sil.nodes:
    sil.nodes[node]['color'] = qpu_bicoloring_a[node]

print("1. PROBLEM (QPU):")
print('GRUP A: \n\t' + '\n\t'.join(list(person for (person, color) in qp
u_bicoloring_a.items() if (color == 0))))
print('GRUP B: \n\t' + '\n\t'.join(list(person for (person, color) in qp
u_bicoloring_a.items() if (color == 1))))
print('DENGESİZ/INSTABİL İLİŞKİLER: \n\t' + '\n\t'.join(list(x + " & " +
y for (x, y) in qpu_imbalance_a.keys())))
```

04/02/2021

TEZ_DOC

1. PROBLEM (QPU):
GRUP A:

EDKCLCLA
 ESDIHDJH
 FUPSMGTE
 GRGUWBGJ
 IKRFBMBDT
 JSEVOJBH
 KEVEIYFM
 KIWVGDHT
 LJHUDUNG
 LJJOJYL
 MNSAGEAC
 MYBYFMLK
 RRTPIOMW
 SUBJWZOO
 TEUQICUF
 VSKEYGVI
 ZCCFYSPB
 ZFOXAGMY

GRUP B:

AWFABBTH
 AXWXFKQO
 BILWCPDD
 BRRLBSVY
 FHUGNOEV
 KJELZABC
 MKVTOATV
 MOYXRFBP
 NRKJALJA
 OCHADGYG
 PVLSNQKX
 QOBRRQLH
 RVBOMUFH
 SAFUFGST
 VAGLFUOB
 VAMAJMIY
 VCCXJKHD

DENGESİZ/INSTABİL İLİŞKİLER:

MOYXRFBP & AXWXFKQO
 MOYXRFBP & SUBJWZOO
 MOYXRFBP & ESDIHDJH
 MOYXRFBP & BRRLBSVY
 MOYXRFBP & LJJOJYL
 MOYXRFBP & AWFABBTH
 MOYXRFBP & MKVTOATV
 MOYXRFBP & EDKCLCLA
 MOYXRFBP & TEUQICUF
 MOYXRFBP & GRGUWBGJ
 MOYXRFBP & OCHADGYG
 MOYXRFBP & JSEVOJBH
 MOYXRFBP & VAGLFUOB
 MOYXRFBP & FHUGNOEV
 MOYXRFBP & FUPSMGTE
 MOYXRFBP & KIWVGDHT
 MOYXRFBP & MNSAGEAC
 MOYXRFBP & MYBYFMLK

04/02/2021

TEZ_DOC

MOYXRFBP & LJHUDUNG
 MOYXRFBP & VSKEYGVI
 MOYXRFBP & ZFOXAGMY
 MOYXRFBP & ZCCFYSPB
 AXWXFKQO & BRLBSVY
 AXWXFKQO & BILWCDD
 AXWXFKQO & SAFUFGST
 AXWXFKQO & KEVEIYFM
 AXWXFKQO & LJHJOJYL
 AXWXFKQO & NRKJALJA
 AXWXFKQO & VAMAJMIY
 AXWXFKQO & MKVTOATV
 AXWXFKQO & EDKCLCLA
 AXWXFKQO & GRGUWBGJ
 AXWXFKQO & IKRFMBDT
 AXWXFKQO & OCHADGYG
 AXWXFKQO & JSEVOJBH
 AXWXFKQO & FUPSMTGE
 AXWXFKQO & RRTPIONMW
 AXWXFKQO & RVBOMUFH
 SUBJWZOO & BRLBSVY
 SUBJWZOO & SAFUFGST
 SUBJWZOO & LJHJOJYL
 SUBJWZOO & AWFABBTH
 SUBJWZOO & VCCXJKHD
 SUBJWZOO & MKVTOATV
 SUBJWZOO & IKRFMBDT
 SUBJWZOO & OCHADGYG
 SUBJWZOO & VAGLFUOB
 SUBJWZOO & FHUGNOEV
 SUBJWZOO & FUPSMTGE
 SUBJWZOO & KJELZABC
 SUBJWZOO & RRTPIONMW
 SUBJWZOO & RVBOMUFH
 SUBJWZOO & ZCCFYSPB
 ESDIHDJH & SAFUFGST
 ESDIHDJH & LJHJOJYL
 ESDIHDJH & TEUQICUF
 ESDIHDJH & IKRFMBDT
 ESDIHDJH & OCHADGYG
 ESDIHDJH & VAGLFUOB
 ESDIHDJH & FHUGNOEV
 ESDIHDJH & FUPSMTGE
 ESDIHDJH & MNSAGEAC
 ESDIHDJH & VSKEYGVI
 ESDIHDJH & RRTPIONMW
 ESDIHDJH & QOBRRQLH
 BRLBSVY & BILWCDD
 BRLBSVY & SAFUFGST
 BRLBSVY & KEVEIYFM
 BRLBSVY & VAMAJMIY
 BRLBSVY & TEUQICUF
 BRLBSVY & GRGUWBGJ
 BRLBSVY & JSEVOJBH
 BRLBSVY & VAGLFUOB
 BRLBSVY & FUPSMTGE
 BRLBSVY & KIWVGDH

localhost:8888/nbconvert/html/Macintosh HD/Users/egedusun/Desktop/Tez/TEZ_DOC.ipynb?download=false

119/146

04/02/2021

TEZ_DOC

BRRRLBSVY & MYBYFMLK
 BRRRLBSVY & VSKEYGVI
 BRRRLBSVY & ZFOXAGMY
 BRRRLBSVY & KJELZABC
 BRRRLBSVY & RVBOMUFH
 BRRRLBSVY & PVLSNQKX
 BRRRLBSVY & ZCCFYSPB
 BRRRLBSVY & QOBRRQLH
 BILWCPDD & KEVEIYFM
 BILWCPDD & LJHOJYL
 BILWCPDD & AWFABTH
 BILWCPDD & MKVTOATV
 BILWCPDD & EDKCLCLIA
 BILWCPDD & GRGUWBGJ
 BILWCPDD & IKRFMBDT
 BILWCPDD & OCHADGYG
 BILWCPDD & JSEVOJBH
 BILWCPDD & VAGLFUOB
 BILWCPDD & FHUGNOEV
 BILWCPDD & KIWVGDHT
 BILWCPDD & MNSAGEAC
 BILWCPDD & MYBYFMLK
 BILWCPDD & VSKEYGVI
 BILWCPDD & ZFOXAGMY
 BILWCPDD & RRTPIOMW
 BILWCPDD & RVBOMUFH
 BILWCPDD & ZCCFYSPB
 BILWCPDD & OOBRRQLH
 SAFUFGST & LJHOJYL
 SAFUFGST & NRKJALJA
 SAFUFGST & VCCXJKHD
 SAFUFGST & MKVTOATV
 SAFUFGST & TEUQICUF
 SAFUFGST & IKRFMBDT
 SAFUFGST & FHUGNOEV
 SAFUFGST & FUPSMTGE
 SAFUFGST & KIWVGDHT
 SAFUFGST & MYBYFMLK
 SAFUFGST & LJHUDUNG
 SAFUFGST & VSKEYGVI
 SAFUFGST & ZFOXAGMY
 SAFUFGST & RRTPIOMW
 KEVEIYFM & LJHOJYL
 KEVEIYFM & VAMAJMIY
 KEVEIYFM & VCCXJKHD
 KEVEIYFM & GRGUWBGJ
 KEVEIYFM & IKRFMBDT
 KEVEIYFM & OCHADGYG
 KEVEIYFM & FHUGNOEV
 KEVEIYFM & MNSAGEAC
 KEVEIYFM & MYBYFMLK
 KEVEIYFM & LJHUDUNG
 KEVEIYFM & KJELZABC
 LJHOJYL & VCCXJKHD
 LJHOJYL & MKVTOATV
 LJHOJYL & OCHADGYG
 LJHOJYL & MYBYFMLK

04/02/2021

TEZ_DOC

LJJHOJYL & LJHUDUNG
 LJJHOJYL & VSKEYGVI
 LJJHOJYL & PVLSNQKX
 LJJHOJYL & ZCCFYSPB
 LJJHOJYL & QOBRRQLH
 AWFABBTH & VCCXJKHD
 AWFABBTH & TEUQICUF
 AWFABBTH & GRGUWBGJ
 AWFABBTH & IKRFMBDT
 AWFABBTH & OCHADGYG
 AWFABBTH & FHUGNOEV
 AWFABBTH & FUPSMTGE
 AWFABBTH & KIWVGDHT
 AWFABBTH & MYBYFMLK
 AWFABBTH & VSKEYGVI
 AWFABBTH & RRTPIOMW
 AWFABBTH & RVBOMUFH
 AWFABBTH & PVLSNQKX
 AWFABBTH & QOBRRQLH
 NRKJALJA & VAMAJMIY
 NRKJALJA & MKVTOATV
 NRKJALJA & EDKCLCLA
 NRKJALJA & JSEVOJBH
 NRKJALJA & VAGLFUOB
 NRKJALJA & KIWVGDHT
 NRKJALJA & VSKEYGVI
 NRKJALJA & RRTPIOMW
 NRKJALJA & RVBOMUFH
 NRKJALJA & PVLSNQKX
 NRKJALJA & ZCCFYSPB
 NRKJALJA & QOBRRQLH
 VAMAJMIY & IKRFMBDT
 VAMAJMIY & OCHADGYG
 VAMAJMIY & FUPSMTGE
 VAMAJMIY & VSKEYGVI
 VAMAJMIY & KJELZABC
 VCCXJKHD & GRGUWBGJ
 VCCXJKHD & IKRFMBDT
 VCCXJKHD & FHUGNOEV
 VCCXJKHD & FUPSMTGE
 VCCXJKHD & KIWVGDHT
 VCCXJKHD & MNSAGEAC
 VCCXJKHD & VSKEYGVI
 VCCXJKHD & ZFOXAGMY
 VCCXJKHD & RRTPIOMW
 VCCXJKHD & PVLSNQKX
 VCCXJKHD & QOBRRQLH
 MKVTOATV & FUPSMTGE
 MKVTOATV & KIWVGDHT
 MKVTOATV & MYBYFMLK
 MKVTOATV & LJHUDUNG
 MKVTOATV & VSKEYGVI
 MKVTOATV & ZFOXAGMY
 MKVTOATV & RRTPIOMW
 MKVTOATV & RVBOMUFH
 MKVTOATV & ZCCFYSPB
 MKVTOATV & QOBRRQLH

04/02/2021

TEZ_DOC

EDKCLCLA & JSEVOJBH
 EDKCLCLA & FHUGNOEV
 EDKCLCLA & KIWVGDHT
 EDKCLCLA & MYBYFMLK
 EDKCLCLA & VSKEYGVI
 EDKCLCLA & ZFOXAGMY
 EDKCLCLA & KJELZABC
 EDKCLCLA & RRTPIOMW
 EDKCLCLA & PVLSNQKX
 TEUQICUF & IKRFBMDT
 TEUQICUF & OCHADGYG
 TEUQICUF & JSEVOJBH
 TEUQICUF & FHUGNOEV
 TEUQICUF & FUPSMTGE
 TEUQICUF & KIWVGDHT
 TEUQICUF & MYBYFMLK
 TEUQICUF & VSKEYGVI
 TEUQICUF & ZCCFYSPB
 GRGUWBGJ & OCHADGYG
 GRGUWBGJ & VAGLFUOB
 GRGUWBGJ & MNSAGEAC
 GRGUWBGJ & LJHUDUNG
 GRGUWBGJ & KJELZABC
 GRGUWBGJ & RRTPIOMW
 IKRFBMDT & JSEVOJBH
 IKRFBMDT & FUPSMTGE
 IKRFBMDT & KIWVGDHT
 IKRFBMDT & LJHUDUNG
 IKRFBMDT & ZFOXAGMY
 IKRFBMDT & KJELZABC
 IKRFBMDT & RRTPIOMW
 IKRFBMDT & PVLSNQKX
 OCHADGYG & VAGLFUOB
 OCHADGYG & KIWVGDHT
 OCHADGYG & MYBYFMLK
 OCHADGYG & LJHUDUNG
 OCHADGYG & VSKEYGVI
 OCHADGYG & PVLSNQKX
 JSEVOJBH & VAGLFUOB
 JSEVOJBH & KIWVGDHT
 JSEVOJBH & MYBYFMLK
 JSEVOJBH & LJHUDUNG
 JSEVOJBH & ZFOXAGMY
 JSEVOJBH & KJELZABC
 JSEVOJBH & RVBOMUFH
 JSEVOJBH & PVLSNQKX
 JSEVOJBH & ZCCFYSPB
 JSEVOJBH & QOBRRQLH
 VAGLFUOB & KIWVGDHT
 VAGLFUOB & MYBYFMLK
 VAGLFUOB & ZFOXAGMY
 VAGLFUOB & RRTPIOMW
 VAGLFUOB & ZCCFYSPB
 FHUGNOEV & KIWVGDHT
 FHUGNOEV & MYBYFMLK
 FHUGNOEV & LJHUDUNG
 FHUGNOEV & VSKEYGVI

04/02/2021

TEZ_DOC

FHUGNOEV & KJELZABC
FHUGNOEV & RRTPIOMW
FHUGNOEV & QOBRRQLH
FUPSMTGE & MNSAGEAC
FUPSMTGE & MYBYFMLK
FUPSMTGE & ZFOXAGMY
FUPSMTGE & KJELZABC
FUPSMTGE & RVBOMUFH
FUPSMTGE & PVLSNQKX
FUPSMTGE & ZCCFYSPB
KIWVGDHT & MNSAGEAC
KIWVGDHT & ZFOXAGMY
KIWVGDHT & KJELZABC
KIWVGDHT & RRTPIOMW
KIWVGDHT & PVLSNQKX
MNSAGEAC & MYBYFMLK
MNSAGEAC & RRTPIOMW
MNSAGEAC & RVBOMUFH
MNSAGEAC & QOBRRQLH
MYBYFMLK & ZFOXAGMY
MYBYFMLK & RRTPIOMW
MYBYFMLK & ZCCFYSPB
MYBYFMLK & QOBRRQLH
LJHUDUNG & ZFOXAGMY
LJHUDUNG & RRTPIOMW
LJHUDUNG & PVLSNQKX
LJHUDUNG & ZCCFYSPB
LJHUDUNG & QOBRRQLH
VSKEYGVI & KJELZABC
VSKEYGVI & RRTPIOMW
VSKEYGVI & RVBOMUFH
VSKEYGVI & PVLSNQKX
VSKEYGVI & ZCCFYSPB
VSKEYGVI & QOBRRQLH
ZFOXAGMY & RVBOMUFH
ZFOXAGMY & QOBRRQLH
KJELZABC & RRTPIOMW
KJELZABC & RVBOMUFH
KJELZABC & ZCCFYSPB
RRTPIOMW & ZCCFYSPB
PVLSNQKX & QOBRRQLH

04/02/2021

TEZ_DOC

```
In [595]: for edge in sil.edges:
    sil.edges[edge]['frustrated'] = edge in hybrid_imbalance_a
for node in sil.nodes:
    sil.nodes[node]['color'] = hybrid_bicoloring_a[node]

print("1. PROBLEM (QPU/CPU):")
print('GRUP A: \n\t' + '\n\t'.join(list(person for (person, color) in hybrid_bicoloring_a.items() if (color == 0))))
print('GRUP B: \n\t' + '\n\t'.join(list(person for (person, color) in hybrid_bicoloring_a.items() if (color == 1))))
print('DENGESİZ/INSTABİL İLİŞKİLER: \n\t' + '\n\t'.join(list(x + " & " + y for (x, y) in hybrid_imbalance_a.keys())))
```

1. PROBLEM (QPU/CPU):
GRUP A:

AWFABBTH
 BRRLBSVY
 EDKCLCLA
 ESDIHDJH
 FHUGNOEV
 GRGUWBGJ
 KEVEIYFM
 KIWVGDHT
 LJHUDUNG
 MKVTOATV
 MYBYFMLK
 NRKJALJA
 OCHADGYG
 SUBJWZOO
 TEUQICUF
 ZCCFYSPB

GRUP B:

AXWXFKQO
 BILWCPDD
 FUPSMTGE
 IKRFMBDT
 JSEVOJBH
 KJELZABC
 LJHHOJYL
 MNSAGEAC
 MOYXRFBP
 PVLSNQKX
 QOBRRQLH
 RRTPIOMW
 RVBOMUFH
 SAFUFGST
 VAGLFUOB
 VAMAJMIY
 VCCXJKHD
 VSKEYGVI
 ZFOXAGMY

DENGESİZ/INSTABİL İLİŞKİLER:

MOYXRFBP & AXWXFKQO
 MOYXRFBP & SUBJWZOO
 MOYXRFBP & ESDIHDJH
 MOYXRFBP & NRKJALJA
 MOYXRFBP & EDKCLCLA
 MOYXRFBP & TEUQICUF
 MOYXRFBP & GRGUWBGJ
 MOYXRFBP & IKRFMBDT
 MOYXRFBP & VAGLFUOB
 MOYXRFBP & KIWVGDHT
 MOYXRFBP & MYBYFMLK
 MOYXRFBP & LJHUDUNG
 MOYXRFBP & RRTPIOMW
 MOYXRFBP & ZCCFYSPB
 AXWXFKQO & BILWCPDD
 AXWXFKQO & SAFUFGST
 AXWXFKQO & KEVEIYFM
 AXWXFKQO & AWFABBTH

04/02/2021

TEZ_DOC

AXWXFKQO & VAMAJMIY
 AXWXFKQO & EDKCLCLA
 AXWXFKQO & GRGUWBGJ
 AXWXFKQO & FHUGNOEV
 AXWXFKQO & MNSAGEAC
 AXWXFKQO & VSKEYGVI
 AXWXFKQO & ZFOXAGMY
 AXWXFKQO & RVBOMUFH
 SUBJWZOO & SAFUFGST
 SUBJWZOO & NRKJALJA
 SUBJWZOO & VCCXJKHD
 SUBJWZOO & JSEVOJBH
 SUBJWZOO & VAGLFUOB
 SUBJWZOO & MNSAGEAC
 SUBJWZOO & VSKEYGVI
 SUBJWZOO & ZFOXAGMY
 SUBJWZOO & KJELZABC
 SUBJWZOO & RVBOMUFH
 SUBJWZOO & ZCCFYSPB
 ESDIHDJH & BRRLBSVY
 ESDIHDJH & SAFUFGST
 ESDIHDJH & AWFABBTH
 ESDIHDJH & NRKJALJA
 ESDIHDJH & MKVTOATV
 ESDIHDJH & TEUQICUF
 ESDIHDJH & JSEVOJBH
 ESDIHDJH & VAGLFUOB
 ESDIHDJH & ZFOXAGMY
 ESDIHDJH & QOBRRQLH
 BRRLBSVY & VCCXJKHD
 BRRLBSVY & EDKCLCLA
 BRRLBSVY & JSEVOJBH
 BRRLBSVY & FUPSMTGE
 BRRLBSVY & LJHUDUNG
 BRRLBSVY & VSKEYGVI
 BRRLBSVY & ZFOXAGMY
 BILWCPDD & KEVEIYFM
 BILWCPDD & NRKJALJA
 BILWCPDD & EDKCLCLA
 BILWCPDD & GRGUWBGJ
 BILWCPDD & VAGLFUOB
 BILWCPDD & FUPSMTGE
 BILWCPDD & KIWVGDHT
 BILWCPDD & MYBYFMLK
 BILWCPDD & RVBOMUFH
 BILWCPDD & ZCCFYSPB
 BILWCPDD & QOBRRQLH
 SAFUFGST & AWFABBTH
 SAFUFGST & VCCXJKHD
 SAFUFGST & TEUQICUF
 SAFUFGST & OCHADGYG
 SAFUFGST & JSEVOJBH
 SAFUFGST & KIWVGDHT
 SAFUFGST & MNSAGEAC
 SAFUFGST & MYBYFMLK
 SAFUFGST & LJHUDUNG
 KEVEIYFM & AWFABBTH

04/02/2021

TEZ_DOC

KEVEIYFM & NRKJALJA
 KEVEIYFM & VAMAJMIY
 KEVEIYFM & VCCXJKHD
 KEVEIYFM & MKVTOATV
 KEVEIYFM & GRGUWBGJ
 KEVEIYFM & JSEVOJBH
 KEVEIYFM & FUPSMTGE
 KEVEIYFM & MYBYFMLK
 KEVEIYFM & LJHUDUNG
 KEVEIYFM & VSKEYGVI
 KEVEIYFM & ZFOXAGMY
 KEVEIYFM & KJELZABC
 KEVEIYFM & RRTPIONW
 LJJHOJYL & VAMAJMIY
 LJJHOJYL & MKVTOATV
 LJJHOJYL & EDKCLCLA
 LJJHOJYL & TEUQICUF
 LJJHOJYL & GRGUWBGJ
 LJJHOJYL & OCHADGYG
 LJJHOJYL & VAGLFUOB
 LJJHOJYL & KIWVGDHT
 LJJHOJYL & VSKEYGVI
 LJJHOJYL & KJELZABC
 LJJHOJYL & RVBOMUFH
 AWFABBTH & VAMAJMIY
 AWFABBTH & EDKCLCLA
 AWFABBTH & IKRFMBDT
 AWFABBTH & OCHADGYG
 AWFABBTH & VAGLFUOB
 AWFABBTH & FHUGNOEV
 AWFABBTH & FUPSMTGE
 AWFABBTH & LJHUDUNG
 AWFABBTH & VSKEYGVI
 AWFABBTH & KJELZABC
 AWFABBTH & RRTPIONW
 AWFABBTH & ZCCFYSPB
 NRKJALJA & VCCXJKHD
 NRKJALJA & MKVTOATV
 NRKJALJA & TEUQICUF
 NRKJALJA & GRGUWBGJ
 NRKJALJA & JSEVOJBH
 NRKJALJA & MYBYFMLK
 NRKJALJA & LJHUDUNG
 NRKJALJA & VSKEYGVI
 NRKJALJA & KJELZABC
 NRKJALJA & RRTPIONW
 VAMAJMIY & MKVTOATV
 VAMAJMIY & JSEVOJBH
 VAMAJMIY & FHUGNOEV
 VAMAJMIY & MNSAGEAC
 VAMAJMIY & ZFOXAGMY
 VAMAJMIY & KJELZABC
 VAMAJMIY & RRTPIONW
 VCCXJKHD & MKVTOATV
 VCCXJKHD & GRGUWBGJ
 VCCXJKHD & OCHADGYG
 VCCXJKHD & JSEVOJBH

04/02/2021

TEZ_DOC

VCCXJKHD & KIWVGDHT
 VCCXJKHD & PVLSNQKX
 VCCXJKHD & QOBRRQLH
 MKVTOATV & EDKCLCLA
 MKVTOATV & TEUQICUF
 MKVTOATV & GRGUWBGJ
 MKVTOATV & VAGLFUOB
 MKVTOATV & FUPSMTGE
 MKVTOATV & VSKEYGVI
 MKVTOATV & ZFOXAGMY
 MKVTOATV & KJELZABC
 MKVTOATV & RRTPIOMW
 MKVTOATV & PVLSNQKX
 EDKCLCLA & IKRFMBDT
 EDKCLCLA & OCHADGYG
 EDKCLCLA & FUPSMTGE
 EDKCLCLA & KIWVGDHT
 EDKCLCLA & MNSAGEAC
 EDKCLCLA & MYBYFMLK
 EDKCLCLA & KJELZABC
 EDKCLCLA & PVLSNQKX
 TEUQICUF & KIWVGDHT
 TEUQICUF & MNSAGEAC
 TEUQICUF & MYBYFMLK
 TEUQICUF & ZFOXAGMY
 TEUQICUF & RRTPIOMW
 TEUQICUF & ZCCFYSPB
 GRGUWBGJ & IKRFMBDT
 GRGUWBGJ & JSEVOJBH
 GRGUWBGJ & VAGLFUOB
 GRGUWBGJ & FHUGNOEV
 GRGUWBGJ & FUPSMTGE
 GRGUWBGJ & LJHUDUNG
 GRGUWBGJ & VSKEYGVI
 GRGUWBGJ & ZFOXAGMY
 GRGUWBGJ & KJELZABC
 IKRFMBDT & JSEVOJBH
 IKRFMBDT & VAGLFUOB
 IKRFMBDT & FUPSMTGE
 IKRFMBDT & MYBYFMLK
 IKRFMBDT & ZFOXAGMY
 IKRFMBDT & RRTPIOMW
 IKRFMBDT & RVBOMUFH
 IKRFMBDT & ZCCFYSPB
 IKRFMBDT & QOBRRQLH
 OCHADGYG & VSKEYGVI
 OCHADGYG & KJELZABC
 OCHADGYG & RVBOMUFH
 OCHADGYG & ZCCFYSPB
 OCHADGYG & QOBRRQLH
 JSEVOJBH & ZFOXAGMY
 VAGLFUOB & FHUGNOEV
 VAGLFUOB & FUPSMTGE
 VAGLFUOB & KIWVGDHT
 VAGLFUOB & MNSAGEAC
 VAGLFUOB & MYBYFMLK
 VAGLFUOB & VSKEYGVI

04/02/2021

TEZ_DOC

```

VAGLFUOB & ZCCFYSPB
FHUGNOEV & VSKEYGVI
FHUGNOEV & RRTPIOMW
FHUGNOEV & RVBOMUFH
FHUGNOEV & PVLSNQKX
FHUGNOEV & ZCCFYSPB
FUPSMGTGE & KIWVGDHT
FUPSMGTGE & MNSAGEAC
FUPSMGTGE & LJHUDUNG
FUPSMGTGE & ZFOXAGMY
FUPSMGTGE & QOBRRQLH
KIWVGDHT & VSKEYGVI
KIWVGDHT & KJELZABC
KIWVGDHT & PVLSNQKX
MNSAGEAC & LJHUDUNG
MNSAGEAC & KJELZABC
MNSAGEAC & RRTPIOMW
MNSAGEAC & PVLSNQKX
MNSAGEAC & ZCCFYSPB
MYBYFMLK & VSKEYGVI
MYBYFMLK & ZCCFYSPB
MYBYFMLK & QOBRRQLH
LJHUDUNG & VSKEYGVI
LJHUDUNG & PVLSNQKX
LJHUDUNG & ZCCFYSPB
LJHUDUNG & QOBRRQLH
VSKEYGVI & RRTPIOMW
ZFOXAGMY & KJELZABC
ZFOXAGMY & PVLSNQKX
ZFOXAGMY & ZCCFYSPB
KJELZABC & RVBOMUFH
KJELZABC & ZCCFYSPB
RRTPIOMW & RVBOMUFH
RRTPIOMW & PVLSNQKX
RRTPIOMW & QOBRRQLH
PVLSNQKX & QOBRRQLH

```

2. Problem

04/02/2021

TEZ_DOC

```
In [594]: for edge in si2.edges:
    si2.edges[edge]['frustrated'] = edge in qpu_imbalance_b
for node in si2.nodes:
    si2.nodes[node]['color'] = qpu_bicoloring_b[node]

print("2. PROBLEM (QPU):")
print('GRUP A: \n\t' + '\n\t'.join(list(person for (person, color) in qp_u_bicoloring_b.items() if (color == 0))))
print('GRUP B: \n\t' + '\n\t'.join(list(person for (person, color) in qp_u_bicoloring_b.items() if (color == 1))))
print('DENGESİZ/INSTABİL İLİŞKİLER: \n\t' + '\n\t'.join(list(x + " & " +
y for (x, y) in qpu_imbalance_b.keys()))))
```

2. PROBLEM (QPU):
GRUP A:

AFIBBHSO
AHVFUTHY
ANJQVHLF
DAZARSPK
GSHGKRQF
NGGEPOQJT
NUYYBHZG
VHNVBXWB
YBXSNOWQ

GRUP B:

ABGOZNAU
BLHGXFHD
HMZJINWY
KBCJPFFR
KJSNDQAN
PPVTEFB
RZSSRPHZ
XGDLWZFC
XIZKOWLP
ZAWUTIVQ
ZRUNMYDJ

DENGESİZ/INSTABİL İLİŞKİLER:

XGDLWZFC & DAZARSPK
XGDLWZFC & AHVFUTHY
XGDLWZFC & ZRUNMYDJ
XGDLWZFC & KJSNDQAN
XGDLWZFC & AFIBBHSO
XGDLWZFC & BLHGXFHD
XGDLWZFC & NUYYBHZG
XGDLWZFC & YBXSNOWQ
XGDLWZFC & XIZKOWLP
XGDLWZFC & ZAWUTIVQ
XGDLWZFC & HMZJINWY
XGDLWZFC & VHNVBXWB
DAZARSPK & AHVFUTHY
DAZARSPK & ABGOZNAU
DAZARSPK & NGGEPOQJT
DAZARSPK & PPVTEFB
DAZARSPK & KBCJPFFR
DAZARSPK & ANJQVHLF
DAZARSPK & VHNVBXWB
AHVFUTHY & ABGOZNAU
AHVFUTHY & RZSSRPHZ
AHVFUTHY & NGGEPOQJT
AHVFUTHY & AFIBBHSO
AHVFUTHY & PPVTEFB
AHVFUTHY & BLHGXFHD
AHVFUTHY & GSHGKRQF
AHVFUTHY & ANJQVHLF
AHVFUTHY & HMZJINWY
ABGOZNAU & ZRUNMYDJ
ABGOZNAU & KJSNDQAN
ABGOZNAU & RZSSRPHZ
ABGOZNAU & NGGEPOQJT
ABGOZNAU & BLHGXFHD

04/02/2021

TEZ_DOC

ABGOZNAU & ANJQVHLF
 ZRUNMYDJ & NGGEPQJT
 ZRUNMYDJ & AFIBBHSD
 ZRUNMYDJ & PPVTEFB
 ZRUNMYDJ & BLHGXFHD
 ZRUNMYDJ & NUYYBHZG
 ZRUNMYDJ & KBCJPFFR
 ZRUNMYDJ & GSHGKRQF
 ZRUNMYDJ & YBXSNOWQ
 ZRUNMYDJ & XIZKOWLP
 ZRUNMYDJ & ZAWUTIVQ
 KJSNDQAN & RZSSRPHZ
 KJSNDQAN & NGGEPQJT
 KJSNDQAN & PPVTEFB
 KJSNDQAN & KBCJPFFR
 RZSSRPHZ & BLHGXFHD
 RZSSRPHZ & XIZKOWLP
 RZSSRPHZ & ANJQVHLF
 RZSSRPHZ & VHNVBXWB
 NGGEPQJT & AFIBBHSD
 NGGEPQJT & BLHGXFHD
 NGGEPQJT & NUYYBHZG
 NGGEPQJT & KBCJPFFR
 NGGEPQJT & ZAWUTIVQ
 NGGEPQJT & HMZJINWY
 AFIBBHSD & NUYYBHZG
 AFIBBHSD & ZAWUTIVQ
 AFIBBHSD & HMZJINWY
 PPVTEFB & KBCJPFFR
 PPVTEFB & GSHGKRQF
 PPVTEFB & YBXSNOWQ
 BLHGXFHD & GSHGKRQF
 BLHGXFHD & ZAWUTIVQ
 BLHGXFHD & ANJQVHLF
 NUYYBHZG & KBCJPFFR
 NUYYBHZG & ZAWUTIVQ
 NUYYBHZG & HMZJINWY
 NUYYBHZG & VHNVBXWB
 KBCJPFFR & YBXSNOWQ
 KBCJPFFR & ZAWUTIVQ
 KBCJPFFR & HMZJINWY
 KBCJPFFR & VHNVBXWB
 YBXSNOWQ & ANJQVHLF
 XIZKOWLP & HMZJINWY
 ZAWUTIVQ & ANJQVHLF
 ANJQVHLF & HMZJINWY
 ANJQVHLF & VHNVBXWB
 HMZJINWY & VHNVBXWB

04/02/2021

TEZ_DOC

```
In [597]: for edge in si2.edges:
    si2.edges[edge]['frustrated'] = edge in hybrid_imbalance_b
for node in si2.nodes:
    si2.nodes[node]['color'] = hybrid_bicoloring_b[node]

print("2. PROBLEM (QPU/CPU):")
print('GRUP A: \n\t' + '\n\t'.join(list(person for (person, color) in hybrid_bicoloring_b.items() if (color == 0))))
print('GRUP B: \n\t' + '\n\t'.join(list(person for (person, color) in hybrid_bicoloring_b.items() if (color == 1))))
print('DENGESİZ/INSTABİL İLİŞKİLER: \n\t' + '\n\t'.join(list(x + " & " + y for (x, y) in hybrid_imbalance_b.keys())))
```

2. PROBLEM (QPU/CPU):

GRUP A:

ABGOZNAU
AHVFUTHY
ANJOVHLF
BLHGXFHD
HMZJINWY
KJSNDQAN
NGGEPQJT
PPVTEFB
RZSSRPHZ
XIZKOWLP
ZAWUTIVQ

GRUP B:

AFIBBHOS
DAZARSPK
GSHGKRQF
KBCJPFFR
NUYYBHZG
VHNVBXB
XGDLWZFC
YBXSNOWQ
ZRUNMYDJ

DENGESİZ/INSTABİL İLİŞKİLER:

XGDLWZFC & AHVFUTHY
XGDLWZFC & ABGOZNAU
XGDLWZFC & ZRUNMYDJ
XGDLWZFC & RZSSRPHZ
XGDLWZFC & PPVTEFB
XGDLWZFC & GSHGKRQF
DAZARSPK & ABGOZNAU
DAZARSPK & ZRUNMYDJ
DAZARSPK & PPVTEFB
DAZARSPK & VHNVBXB
AHVFUTHY & KJSNDQAN
AHVFUTHY & NGGEPQJT
AHVFUTHY & NUYYBHZG
AHVFUTHY & YBXSNOWQ
AHVFUTHY & XIZKOWLP
AHVFUTHY & ZAWUTIVQ
AHVFUTHY & ANJQVHLF
AHVFUTHY & VHNVBXB
ABGOZNAU & KJSNDQAN
ABGOZNAU & RZSSRPHZ
ABGOZNAU & BLHGXFHD
ABGOZNAU & KBCJPFFR
ZRUNMYDJ & KJSNDQAN
ZRUNMYDJ & RZSSRPHZ
ZRUNMYDJ & NGGEPQJT
ZRUNMYDJ & KBCJPFFR
ZRUNMYDJ & HMZJINWY
ZRUNMYDJ & VHNVBXB
KJSNDQAN & RZSSRPHZ
KJSNDQAN & PPVTEFB
KJSNDQAN & ANJQVHLF
RZSSRPHZ & NGGEPQJT
RZSSRPHZ & BLHGXFHD

04/02/2021

TEZ_DOC

RZSSRPHZ & KBCJPFFR
RZSSRPHZ & XIZKOWLP
RZSSRPHZ & VHNVBXWB
NGGEPOQJT & PPVTEFB
NGGEPOQJT & KBCJPFFR
NGGEPOQJT & GSHGKRQF
NGGEPOQJT & YBXSNOWQ
NGGEPOQJT & XIZKOWLP
NGGEPOQJT & VHNVBXWB
AFIBBHSD & NUYYBHZG
AFIBBHSD & KBCJPFFR
AFIBBHSD & ZAWUTIVQ
AFIBBHSD & ANJQVHLF
AFIBBHSD & HMZJINWY
PPVTEFB & GSHGKRQF
PPVTEFB & YBXSNOWQ
PPVTEFB & ANJQVHLF
BLHGKFHD & KBCJPFFR
BLHGKFHD & GSHGKRQF
BLHGKFHD & ZAWUTIVQ
NUYYBHZG & ZAWUTIVQ
NUYYBHZG & ANJQVHLF
NUYYBHZG & HMZJINWY
NUYYBHZG & VHNVBXWB
KBCJPFFR & GSHGKRQF
KBCJPFFR & XIZKOWLP
GSHGKRQF & ANJQVHLF
XIZKOWLP & ANJQVHLF
XIZKOWLP & HMZJINWY
HMZJINWY & VHNVBXWB

3. Problem

04/02/2021

TEZ_DOC

```
In [598]: for edge in si3.edges:
    si3.edges[edge]['frustrated'] = edge in qpu_imbalance_c
for node in si3.nodes:
    si3.nodes[node]['color'] = qpu_bicoloring_c[node]

print("3. PROBLEM (QPU):")
print('GRUP A: \n\t' + '\n\t'.join(list(person for (person, color) in qp
u_bicoloring_c.items() if (color == 0))))
print('GRUP B: \n\t' + '\n\t'.join(list(person for (person, color) in qp
u_bicoloring_c.items() if (color == 1))))
print('DENGESİZ/INSTABİL İLİŞKİLER: \n\t' + '\n\t'.join(list(x + " & " +
y for (x, y) in qpu_imbalance_c.keys()))))
```

3. PROBLEM (QPU):
GRUP A:

BHVEAPRL
CYCWPWLY
EDCMWUFO
EHRLXQCK
IAMVGBLQ
IBJGPPLY
MZTDNRSY
QISKCZAM
QZBUYHLX
SSRVFBMF
TOEMAWHU
VCVBBZMO
VOYIHDA
WNWBWLXD
ZAJLVQLL
ZUUKMSLS

GRUP B:

BPBSCVGO
CFSZAMZC
CPTVVBFT
CZHSWOIZ
FIAHDDJC
JRAWRBZA
KVCGBBT
LDHLLNWQ
NIZLNAGM
PKJFPRNS
RTTYCLPN
SMWBGJOB
TMOGLXZE
YOUQDZFH

DENGESİZ/INSTABİL İLİŞKİLER:

LDHLLNWQ & MZTDNRSY
LDHLLNWQ & KVCGBBT
LDHLLNWQ & TOEMAWHU
LDHLLNWQ & QZBUYHLX
LDHLLNWQ & SSRVFBMW
LDHLLNWQ & CFSZAMZC
LDHLLNWQ & IBJGPPLY
LDHLLNWQ & ZAJLVQLL
LDHLLNWQ & NIZLNAGM
LDHLLNWQ & PKJFPRNS
LDHLLNWQ & VOYIHDA
LDHLLNWQ & FIAHDDJC
LDHLLNWQ & VCVBBZMO
LDHLLNWQ & CZHSWOIZ
LDHLLNWQ & JRAWRBZA
LDHLLNWQ & ZUUKMSLS
IAMVGBLQ & MZTDNRSY
IAMVGBLQ & KVCGBBT
IAMVGBLQ & RTTYCLPN
IAMVGBLQ & SSRVFBMW
IAMVGBLQ & EDCMWUFO
IAMVGBLQ & CYCWPWLY
IAMVGBLQ & WNBWLXD

04/02/2021

TEZ_DOC

IAMVGBLQ & BPBSCVGO
 IAMVGBLQ & SMWBGJOB
 IAMVGBLQ & ZUUKMSLS
 QISKCZAM & MZTDNRSY
 QISKCZAM & KVCGBTAK
 QISKCZAM & TOEMAWHU
 QISKCZAM & Q2BUYHLX
 QISKCZAM & CPTVVBF
 QISKCZAM & NIZLNAGM
 QISKCZAM & PKJFPRNS
 QISKCZAM & TMGLXZE
 QISKCZAM & WNBWBLXD
 QISKCZAM & VCVBBZMO
 QISKCZAM & CZHSWOIZ
 MZTDNRSY & TOEMAWHU
 MZTDNRSY & Q2BUYHLX
 MZTDNRSY & SSRVFBMF
 MZTDNRSY & CPTVVBF
 MZTDNRSY & ZAJLVQLL
 MZTDNRSY & NIZLNAGM
 MZTDNRSY & CYCWPPLY
 MZTDNRSY & TMGLXZE
 MZTDNRSY & VCVBBZMO
 MZTDNRSY & CZHSWOIZ
 MZTDNRSY & JRAWRBZA
 YOUQDZFH & Q2BUYHLX
 YOUQDZFH & RTTYCLPN
 YOUQDZFH & EDCMWUFO
 YOUQDZFH & CFSZAMZC
 YOUQDZFH & CPTVVBF
 YOUQDZFH & NIZLNAGM
 YOUQDZFH & WNBWBLXD
 YOUQDZFH & VCVBBZMO
 YOUQDZFH & CZHSWOIZ
 YOUQDZFH & BPBSCVGO
 KVCGBTAK & BHVEAPRL
 KVCGBTAK & SSRVFBMF
 KVCGBTAK & CFSZAMZC
 KVCGBTAK & CPTVVBF
 KVCGBTAK & FIAHDDJC
 KVCGBTAK & CZHSWOIZ
 KVCGBTAK & JRAWRBZA
 KVCGBTAK & SMWBGJOB
 TOEMAWHU & SSRVFBMF
 TOEMAWHU & EDCMWUFO
 TOEMAWHU & CPTVVBF
 TOEMAWHU & IBJGPPLY
 TOEMAWHU & NIZLNAGM
 TOEMAWHU & PKJFPRNS
 TOEMAWHU & TMGLXZE
 TOEMAWHU & VCVBBZMO
 TOEMAWHU & CZHSWOIZ
 TOEMAWHU & SMWBGJOB
 TOEMAWHU & EHRLXQCK
 BHVEAPRL & Q2BUYHLX
 BHVEAPRL & RTTYCLPN
 BHVEAPRL & SSRVFBMF

localhost:8888/nbconvert/html/Macintosh HD/Users/egedusun/Desktop/Tez/TEZ_DOC.ipynb?download=false

138/146

04/02/2021

TEZ_DOC

BHVEAPRL & NIZLNAGM
 BHVEAPRL & PKJFPRNS
 BHVEAPRL & VOYIHDVA
 BHVEAPRL & CYCWPWLY
 BHVEAPRL & WNBWBLXD
 BHVEAPRL & FIAHDDJC
 BHVEAPRL & JRAWRBZA
 BHVEAPRL & BPBSCVGO
 BHVEAPRL & ZUUKMSLS
 QZBUYHLX & RTTYCLPN
 QZBUYHLX & EDCMWUFO
 QZBUYHLX & CFSZAMZC
 QZBUYHLX & CPTVVBF
 QZBUYHLX & ZAJLVQLL
 QZBUYHLX & NIZLNAGM
 QZBUYHLX & PKJFPRNS
 QZBUYHLX & CYCWPWLY
 QZBUYHLX & FIAHDDJC
 QZBUYHLX & VCVBBZMO
 QZBUYHLX & JRAWRBZA
 QZBUYHLX & BPBSCVGO
 QZBUYHLX & SMWBGJOB
 QZBUYHLX & ZUUKMSLS
 QZBUYHLX & EHRLXQCK
 RTTYCLPN & IBJGPFLY
 RTTYCLPN & NIZLNAGM
 RTTYCLPN & PKJFPRNS
 RTTYCLPN & CYCWPWLY
 RTTYCLPN & WNBWBLXD
 RTTYCLPN & VCVBBZMO
 RTTYCLPN & CZHSWOIZ
 RTTYCLPN & JRAWRBZA
 RTTYCLPN & BPBSCVGO
 RTTYCLPN & SMWBGJOB
 RTTYCLPN & ZUUKMSLS
 SSRVFBNF & EDCMWUFO
 SSRVFBNF & IBJGPFLY
 SSRVFBNF & NIZLNAGM
 SSRVFBNF & CZHSWOIZ
 SSRVFBNF & JRAWRBZA
 SSRVFBNF & BPBSCVGO
 SSRVFBNF & EHRLXQCK
 EDCMWUFO & CPTVVBF
 EDCMWUFO & VOYIHDVA
 EDCMWUFO & WNBWBLXD
 EDCMWUFO & FIAHDDJC
 EDCMWUFO & VCVBBZMO
 EDCMWUFO & JRAWRBZA
 EDCMWUFO & BPBSCVGO
 EDCMWUFO & ZUUKMSLS
 CFSZAMZC & CPTVVBF
 CFSZAMZC & VOYIHDVA
 CFSZAMZC & FIAHDDJC
 CPTVVBF & ZAJLVQLL
 CPTVVBF & NIZLNAGM
 CPTVVBF & PKJFPRNS
 CPTVVBF & VOYIHDVA

04/02/2021

TEZ_DOC

CPTVVBFT & TMOGLXZE
 CPTVVBFT & VCVBBZMO
 CPTVVBFT & CZHSWOIZ
 CPTVVBFT & BPBSCVGO
 IBJGPPLY & VOYIHDVA
 IBJGPPLY & CYCWPWLY
 IBJGPPLY & TMOGLXZE
 IBJGPPLY & WNBWBLXD
 IBJGPPLY & VCVBBZMO
 IBJGPPLY & JRAWRBZA
 IBJGPPLY & BPBSCVGO
 IBJGPPLY & ZUUKMSLS
 ZAJLVQLL & TMOGLXZE
 ZAJLVQLL & WNBWBLXD
 ZAJLVQLL & VCVBBZMO
 ZAJLVQLL & JRAWRBZA
 ZAJLVQLL & SMWBGJOB
 ZAJLVQLL & ZUUKMSLS
 NIZLNAGM & TMOGLXZE
 NIZLNAGM & FIAHDDJC
 NIZLNAGM & VCVBBZMO
 NIZLNAGM & JRAWRBZA
 NIZLNAGM & BPBSCVGO
 PKJFPRNS & CYCWPWLY
 PKJFPRNS & FIAHDDJC
 PKJFPRNS & VCVBBZMO
 PKJFPRNS & CZHSWOIZ
 PKJFPRNS & BPBSCVGO
 PKJFPRNS & ZUUKMSLS
 VOYIHDVA & CYCWPWLY
 VOYIHDVA & FIAHDDJC
 VOYIHDVA & VCVBBZMO
 VOYIHDVA & JRAWREZA
 VOYIHDVA & SMWBGJOB
 VOYIHDVA & ZUUKMSLS
 CYCWPWLY & TMOGLXZE
 CYCWPWLY & FIAHDDJC
 CYCWPWLY & VCVBBZMO
 CYCWPWLY & JRAWRBZA
 CYCWPWLY & BPBSCVGO
 CYCWPWLY & ZUUKMSLS
 CYCWPWLY & EHRLXQCK
 TMOGLXZE & VCVBBZMO
 TMOGLXZE & CZHSWOIZ
 TMOGLXZE & SMWBGJOB
 WNBWBLXD & CZHSWOIZ
 WNBWBLXD & BPBSCVGO
 FIAHDDJC & VCVBBZMO
 FIAHDDJC & BPBSCVGO
 FIAHDDJC & SMWBGJOB
 FIAHDDJC & ZUUKMSLS
 VCVBBZMO & ZUUKMSLS
 VCVBBZMO & EHRLXQCK
 CZHSWOIZ & JRAWRBZA
 CZHSWOIZ & BPBSCVGO
 CZHSWOIZ & SMWBGJOB
 CZHSWOIZ & EHRLXQCK

04/02/2021

TEZ_DOC

JRAWRBZA & BPBSCVGO
JRAWRBZA & SMWBGJOB
JRAWRBZA & ZUUKMSLS
JRAWRBZA & EHRLXQCK
BPBSCVGO & ZUUKMSLS
BPBSCVGO & EHRLXQCK
SMWBGJOB & EHRLXQCK
ZUUKMSLS & EHRLXQCK

04/02/2021

TEZ_DOC

```
In [599]: for edge in si3.edges:
    si3.edges[edge]['frustrated'] = edge in hybrid_imbalance_c
for node in si3.nodes:
    si3.nodes[node]['color'] = hybrid_bicoloring_c[node]

print("3. PROBLEM (QPU/CPU):")
print('GRUP A: \n\t' + '\n\t'.join(list(person for (person, color) in hybrid_bicoloring_c.items() if (color == 0))))
print('GRUP B: \n\t' + '\n\t'.join(list(person for (person, color) in hybrid_bicoloring_c.items() if (color == 1))))
print('DENGESİZ/INSTABİL İLİŞKİLER: \n\t' + '\n\t'.join(list(x + " & " +
y for (x, y) in hybrid_imbalance_c.keys())))
```

04/02/2021

TEZ_DOC

3. PROBLEM (QPU/CPU):**GRUP A:**

BPBSCVGO
 CFSZAMZC
 CYCWPWLY
 KVCGBTK
 LDHLLNWQ
 NIZLNAGM
 PKJFPRNS
 QZBUYHLX
 RTTYCLPN
 SMWBGJOB
 SSRVFBMF
 TMOGLXZE
 TOEMAWHU
 VCVBBZMO
 WNBWBLXD
 YOUQDZFH
 ZUUKMSLS

GRUP B:

BHVEAPRL
 CPTVVBFT
 CZHSWOIZ
 EDCMWUFO
 EHRLXQCK
 FIAHDDJC
 IAMVGBLQ
 IBJGPPLY
 JRAWRBZA
 MZTDNRSY
 QISKCZAM
 VOYIHDA
 ZAJLVQLL

DENGESİZ/INSTABİL İLİŞKİLER:

LDHLLNWQ & MZTDNRSY
 LDHLLNWQ & KVCGBTK
 LDHLLNWQ & CFSZAMZC
 LDHLLNWQ & CPTVVBFT
 LDHLLNWQ & IBJGPPLY
 LDHLLNWQ & ZAJLVQLL
 LDHLLNWQ & NIZLNAGM
 LDHLLNWQ & PKJFPRNS
 LDHLLNWQ & VOYIHDA
 LDHLLNWQ & CYCWPWLY
 LDHLLNWQ & WNBWBLXD
 IAMVGBLQ & MZTDNRSY
 IAMVGBLQ & KVCGBTK
 IAMVGBLQ & TOEMAWHU
 IAMVGBLQ & QZBUYHLX
 IAMVGBLQ & RTTYCLPN
 IAMVGBLQ & EDCMWUFO
 IAMVGBLQ & CPTVVBFT
 IAMVGBLQ & FIAHDDJC
 IAMVGBLQ & VCVBBZMO
 IAMVGBLQ & CZHSWOIZ
 IAMVGBLQ & JRAWRBZA
 IAMVGBLQ & BPBSCVGO

04/02/2021

TEZ_DOC

IAMVGBLQ & SMWBGJOB
 QISKCZAM & MZTDNRSY
 QISKCZAM & KVCGBTAK
 QISKCZAM & SSRVFEMF
 QISKCZAM & NIZLNAGM
 QISKCZAM & PKJFPRLS
 QISKCZAM & CYCWPWLY
 QISKCZAM & TMGLXZE
 QISKCZAM & FIAHDDJC
 QISKCZAM & JRAWRBZA
 QISKCZAM & ZUUKMSLS
 MZTDNRSY & ZAJLVQLL
 MZTDNRSY & NIZLNAGM
 MZTDNRSY & TMGLXZE
 MZTDNRSY & WNBWBLXD
 MZTDNRSY & FIAHDDJC
 MZTDNRSY & ZUUKMSLS
 YOUDZFH & TOEMAWHU
 YOUDZFH & RTTYCLPN
 YOUDZFH & SSRVFEMF
 YOUDZFH & EDCMWUFO
 YOUDZFH & CFSZAMZC
 YOUDZFH & NIZLNAGM
 YOUDZFH & CYCWPWLY
 YOUDZFH & FIAHDDJC
 YOUDZFH & JRAWRBZA
 YOUDZFH & BPBSCVGO
 YOUDZFH & ZUUKMSLS
 KVCGBTAK & TOEMAWHU
 KVCGBTAK & BHVEAPRL
 KVCGBTAK & QZBUYHLX
 KVCGBTAK & CFSZAMZC
 KVCGBTAK & CYCWPWLY
 KVCGBTAK & WNBWBLXD
 KVCGBTAK & VCVBBZMO
 KVCGBTAK & SMWBGJOB
 KVCGBTAK & ZUUKMSLS
 TOEMAWHU & BHVEAPRL
 TOEMAWHU & RTTYCLPN
 TOEMAWHU & SSRVFEMF
 TOEMAWHU & CFSZAMZC
 TOEMAWHU & CPTVVBF
 TOEMAWHU & ZAJLVQLL
 TOEMAWHU & VOYIHDA
 TOEMAWHU & VCVBBZMO
 TOEMAWHU & CZHSWOIZ
 TOEMAWHU & BPBSCVGO
 BHVEAPRL & RTTYCLPN
 BHVEAPRL & CPTVVBF
 BHVEAPRL & NIZLNAGM
 BHVEAPRL & PKJFPRLS
 BHVEAPRL & VOYIHDA
 BHVEAPRL & VCVBBZMO
 BHVEAPRL & CZHSWOIZ
 BHVEAPRL & BPBSCVGO
 QZBUYHLX & CPTVVBF
 QZBUYHLX & IBJGPPLY

04/02/2021

TEZ_DOC

QZBUYHLX & VOYIHDVA
 QZBUYHLX & CYCWPWLY
 QZBUYHLX & TMGLXZE
 QZBUYHLX & FIAHDDJC
 QZBUYHLX & VCVBBZMO
 QZBUYHLX & JRAWRBZA
 QZBUYHLX & ZUUKMSLS
 RTTYCLPN & SSRVFBMF
 RTTYCLPN & CPTVVBF
 RTTYCLPN & IBJGPPLY
 RTTYCLPN & NIZLNAGM
 RTTYCLPN & PKJFPRNS
 RTTYCLPN & FIAHDDJC
 RTTYCLPN & BPBSCVGO
 RTTYCLPN & SMWBGJOB
 SSRVFBMF & CFSZAMZC
 SSRVFBMF & ZAJLVQLL
 SSRVFBMF & PKJFPRNS
 SSRVFBMF & VOYIHDVA
 SSRVFBMF & TMGLXZE
 SSRVFBMF & CZHSWOIZ
 SSRVFBMF & JRAWRBZA
 SSRVFBMF & SMWBGJOB
 EDCMWUFO & VOYIHDVA
 EDCMWUFO & CYCWPWLY
 EDCMWUFO & CZHSWOIZ
 EDCMWUFO & BPBSCVGO
 CFSZAMZC & VOYIHDVA
 CFSZAMZC & CYCWPWLY
 CFSZAMZC & WNBWBLXD
 CFSZAMZC & VCVBBZMO
 CFSZAMZC & CZHSWOIZ
 CFSZAMZC & JRAWREZA
 CFSZAMZC & ZUUKMSLS
 CPTVVBF & IBJGPPLY
 CPTVVBF & VCVBBZMO
 CPTVVBF & CZHSWOIZ
 CPTVVBF & SMWBGJOB
 CPTVVBF & EHRLXQCK
 IBJGPPLY & VOYIHDVA
 IBJGPPLY & TMGLXZE
 IBJGPPLY & FIAHDDJC
 IBJGPPLY & CZHSWOIZ
 IBJGPPLY & BPBSCVGO
 ZAJLVQLL & CYCWPWLY
 ZAJLVQLL & TMGLXZE
 ZAJLVQLL & FIAHDDJC
 ZAJLVQLL & CZHSWOIZ
 ZAJLVQLL & SMWBGJOB
 NIZLNAGM & CYCWPWLY
 NIZLNAGM & TMGLXZE
 NIZLNAGM & WNBWBLXD
 NIZLNAGM & CZHSWOIZ
 NIZLNAGM & BPBSCVGO
 NIZLNAGM & ZUUKMSLS
 PKJFPRNS & WNBWBLXD
 PKJFPRNS & JRAWRBZA

04/02/2021

TEZ_DOC

```
PKJFPRNS & BPBSCVGO
VOYIHDVA & WNBWBLXD
VOYIHDVA & CZHSWOIZ
VOYIHDVA & SMWBGJOB
CYCWPWLY & FIAHDDJC
CYCWPWLY & VCVBBZMO
CYCWPWLY & JRAWRBZA
CYCWPWLY & SMWBGJOB
CYCWPWLY & ZUUKMSLS
TMOGLXZE & WNBWBLXD
TMOGLXZE & FIAHDDJC
TMOGLXZE & JRAWRBZA
TMOGLXZE & SMWBGJOB
TMOGLXZE & ZUUKMSLS
WNBWBLXD & CZHSWOIZ
WNBWBLXD & SMWBGJOB
WNBWBLXD & EHRLXQCK
FIAHDDJC & VCVBBZMO
FIAHDDJC & ZUUKMSLS
FIAHDDJC & EHRLXQCK
VCVBBZMO & BPBSCVGO
VCVBBZMO & SMWBGJOB
VCVBBZMO & ZUUKMSLS
CZHSWOIZ & JRAWRBZA
JRAWRBZA & ZUUKMSLS
BPBSCVGO & EHRLXQCK
SMWBGJOB & ZUUKMSLS
SMWBGJOB & EHRLXQCK
```

In []: