Int. J. Adv. Eng. Pure Sci. 2026, 38(1): <91-104>
DOI: 10.7240/jeps.1752589

RESEARCH ARTICLE / ARAŞTIRMA MAKALESİ

Kuantum Yaklaşık Optimizasyon Algoritması İçin Hamilton İşlemcisi İnşası Yöntemlerinin Karşılaştırmalı Analizi

Comparative Analysis of Hamilton Construction Methods for the Quantum Approximate Optimization Algorithm

Saba Arife BOZPOLAT [1], Özlem SALEHİ KÖKEN [2,3]

[1]*Marmara Üniversitesi, Fen Fakültesi, Fizik Bölümü, İstanbul, Türkiye*

[2]*Sabancı Üniversitesi, Mühendislik ve Doğa Bilimleri Fakültesi, Bilgisayar Bilimi ve Mühendisliği Bölümü, İstanbul, Türkiye*

[3]*Algorithmiq Ltd, Helsinki, Finland*

Öz

Kuantum hesaplama, karmaşık optimizasyon problemlerini çözmede devrim niteliğinde bir potansiyele sahiptir. Kuantum yaklaşık optimizasyon algoritması (QAOA), bu alandaki en önemli yöntemlerden biridir. Çalışma özellikle iki parçalı ağırlıklı graf eşleştirme problemine odaklanmaktadır. İlk yaklaşım, kısıtların ihlal edilmesi durumunda amaç fonksiyonunu artıran ceza yöntemini kullanmaktadır. İlk yaklaşım, kısıtlar ihlal edildiğinde amaç fonksiyonunu artıran ceza yöntemini kullanmaktadır. İkinci yaklaşım ise, kısıtları doğrudan Boolean fonksiyonları olarak ifade ederek Hamilton işlemcisi oluşturan Hadfield'ın Boolean temsiline dayanmaktadır. Elde edilen sonuçlar, Hadfield'ın Boolean temsili kullanılarak inşa edilen Hamilton işlemcisinin optimal sonuca ulaşma açısından daha etkili bir performans sergilediğini göstermektedir. Bu bulgu, kısıtlı kombinatorik optimizasyon problemlerinde Hadfield'ın Boolean temsil yönteminin potansiyelini vurgulamaktadır.
Anahtar Kelimeler: QAOA, Hamilton işlemcisi, iki parçalı graf, eşleştirme problemi, ceza yöntemi

Abstract

Quantum computing has the potential to revolutionize the solution of complex optimization problems. Quantum approximate optimization algorithm (QAOA) is one of the most prominent methods in this field. This study compares two different approaches to Hamiltonian construction within the context of QAOA. Specifically, it focuses on the weighted bipartite graph matching problem. The first approach employs the penalty method, which increases the objective function when constraints are violated. The second approach is based on Hadfield's Boolean representation, which constructs Hamiltonians directly from constraints expressed as Boolean functions. The results obtained indicate that the Hamiltonian constructed using Hadfield's method demonstrates a more effective performance in reaching optimal outcomes. This finding underscores the potential of Hadfield's method in constrained combinatorial optimization problems.
Keywords: QAOA, Hamiltonian, weighted bipartite graph, matching problem, penalty method

I. GİRİŞ

Kuantum yaklaşık optimizasyon algoritması (QAOA - quantum approximate optimization algorithm) [1], kombinatorik optimizasyon problemlerini klasik ve kuantum yaklaşımların bir kombinasyonuyla, yani hibrit olarak çözmek üzere tasarlanmış bir gürültülü orta ölçekli kuantum (NISQ) [2] algoritmasıdır. Bu algoritma, adyabatik kuantum hesaplamadan [3] esinlenilerek kuantum devre modeli için tasarlanmıştır. Amaç Hamilton işlemcisi (Hamiltonian), çözülmek istenen problemi kodlayan bir Ising Hamilton işlemcisi iken, karıştırıcı Hamilton işlemcisi olası çözümler arası genlik geçişini sağlar. QAOA algoritmasında, bu iki Hamilton işlemcisi dönüşümlü olarak uygulanır. Uygulama süreleri, parametrik olarak tanımlanır ve bu parametreler klasik optimizasyon yöntemleriyle optimize edilir. Geçtiğimiz yıllarda QAOA algoritmasının pek çok varyantı tasarlanmıştır [4-8] ve QAOA algoritması teorik olarak incelenmiştir [9-12].

Bir kombinatorik optimizasyon probleminin QAOA ile çözümü için, probleme özgü Ising Hamilton

Sorumlu Yazar: ÖZLEM SALEHİ KÖKEN, **Tel:** 0216 4839601, **E-posta:** ozlemsalehi@gmail.com

Gönderilme: 28.07.2025, **Düzenleme:** 11.12.2025, **Kabul:** 28.01.2026

işlemcisinin tasarlanması gerekir [13]. Bu noktada, genellikle problem öncelikle bir kuadratik kısıtsız ikili optimizasyon (QUBO) problemi şeklinde ifade edilir [14]. Bu formülasyonun avantajı, ikili değişkenler üzerinde tanımlandığından problemin daha kolay formüle dökülebilmesi, ve doğrusal programlama yöntemleriyle elde edilen kısıt içeren formülasyonların QUBO formuna kolayca çevrilebilmesidir. Pek çok optimizasyon problemi doğası gereği kısıt içerdiğinden bu son nokta önemlidir. Sonrasında, elde edilen QUBO formülasyonu, basit bir şekilde Ising modeline dönüştürülebilir. Ancak kombinatorik optimizasyon problemleri genellikle pek çok kısıt içerirler. Kısıtlı problemleri QUBO formuna çevirmek için kullanılan en popüler yöntemlerden biri ceza yöntemidir [15]. Bu yöntemde, kısıtlar ihlal edildikleri durumda amaç fonksiyonun değerini artıracak şekilde amaç fonksiyonuna eklenir. İhlal edilmedikleri durumda amaç fonksiyonu aynı kalır.

Kısıtları olan bir optimizasyon probleminin Hamilton işlemcisini elde etmek için bir diğer yöntem, QUBO formülasyonuna geçmeden doğrudan Hamilton işlemcisi inşasıdır. Hadfield vd., 2021'de yayınlanan makalelerinde [16], Boolean ve reel fonksiyonlar için Boolean temsilları ve Fourier analizi kullanarak Hamilton işlemcisi inşası önermiştir. Bu yöntem sayesinde, kısıtlar Boolean fonksiyonlar olarak ifade edilerek doğrudan Hamilton işlemcisi elde edilebilir.
Bu makalede, Hamilton işlemcisi inşasında ceza yöntemi ile Hadfield'ın Boolean temsil yöntemleri, iki parçalı ağrılıklı graf üzerinde eşleştirme probleminin QAOA algoritması ile çözümü için karşılaştırılmaktadır. İki parçalı ağırlıklı graf üzerinde eşleşme probleminin, böbrek nakli eşleştirmesi [17], işçi-iş eşleştirme problemi [18], müşteri-reklam eşleşmesi [19] gibi pek çok farklı alanda uygulamaları vardır. Pek çok diğer optimizasyon problemi ile temelde benzer matematiksel yapıya sahiptir. Bu problem için öncelikle iki farklı yöntem kullanılarak Hamilton işlemcisi inşası yapılmıştır. Sonrasında elde edilen Hamilton işlemcileri kullanılarak QAOA devreleri elde edilmiştir. Bu devreler, simülatörde çalıştırılarak elde edilen sonuçlar optimal sonuca yakınlık ve fizibilite açısından incelenmiştir. Sonuçlar, Hadfield'ın Boolean temsil yöntemi ile elde edilen Hamilton işlemcisi kullanıldığında QAOA algoritmasının daha başarılı çalıştığını göstermiştir.

Bu çalışmanın katkıları üç başlık altında özetlenebilir. (i) İlk olarak, önerilen iki farklı inşa paradigması k-lokallik ve kapı sayısı açısından sistematik biçimde analiz edilerek kuramsal ve pratik karmaşıklıkları karşılaştırılmıştır. (ii) İkinci olarak, üç farklı senaryo ve üç farklı optimizatörün değerlendirilmesiyle sağlam bir kıyas sunulmuştur. (iii) Son olarak, özellikle NISQ cihazlarını hedefleyerek düşük p rejiminde elde edilen bulgular ışığında, pratik uygulamalara yönelik somut öneriler ortaya konmuştur.

Makalenin ikinci bölümünde, konuyla ilgili gerekli arka plan aktarılmıştır. Üçüncü bölümde, iki parçalı graf üzerinde eşleştirme problemi matematiksel olarak tanımlanmış ve Hamilton işlemcisi inşaları göstermiştir. Dördüncü bölümde QAOA algoritması iki farklı Hamilton işlemcisi için çalıştırılarak sonuçlar derlenmiştir. Beşinci bölümde ise çalışma sonuçlandırılmıştır.

II. MATERYAL VE METOD

Bu bölümde, makale boyunca kullanılacak olan temel kavramları tanıtacağız. Sırasıyla, 0-1 programlama formülasyonu, QUBO formülasyonu, Ising Modeli, ve QAOA algoritmasından bahsedeceğiz.

2.1. 0-1 Programlama formülasyonu

0-1 programlama, karar değişkenlerinin yalnızca iki değerden birini alabileceği, yani 0 veya 1 değerlerini alabileceği bir doğrusal programlama türüdür. Minimize edilmek istenen bir hedef fonksiyonu ve kısıtlardan oluşur.

$$min. \sum_{i=1}^{n} x_i \quad \textbf{(1)}$$

$$s.t. \sum_{i=1}^{n} a_{i,j} x_i \leq b_j \quad \forall j = 1,2,\dots,m \quad \textbf{(2)}$$

$$x_i \in \{0,1\} \; i = 1,2,\dots,n \quad \textbf{(3)}$$

0-1 programlama, tam sayılı programlamanın değişkenlerin yalnızca 0-1 üzerinde tanımladığı özel halidir. Bu nedenle tam sayılı programları çözmek için kullanılan tüm yöntemler 0-1 programlama için de kullanılabilir [20].

2.2. QUBO formülasyonu

Kuadratik kısıtsız ikili optimizasyon (QUBO-Quadratic Unconstrained Binary Optimization) formülasyonu, çeşitli kombinatoryal optimizasyon problemlerini modellemek için kullanılan bir yöntemdir [14]. QUBO formülasyonunda, karar değişkenleri sadece iki değer alabilir. Minimize edilmek istenen bir hedef fonksiyonu vardır, kısıt bulunmamaktadır.

Matematiksel olarak QUBO formülasyonu aşağıdaki şekilde ifade edilir. Eşitlik (4)'te $x_i \in \{0,1\}$ olmak üzere ikili değişkenler, $f(x)$ ise minimize etmek istediğimiz fonksiyondur. Bu formülasyonda kısıt bulunmamaktadır.

$$f(x) = \sum_{i \leq j} Q_{i,j} x_i x_j \quad \textbf{(4)}$$

Verilen bir 0-1 programlama formülasyonu, QUBO formülasyonuna dönüştürülebilir. Bunun için kullanılacak metodlardan biri ceza yöntemidir [15]. Bu yöntemde, kısıtlar $f(x)$ fonksiyonuna aşağıda anlatılacağı üzere eklenerek, kısıt ihlal edildiğinde $f(x)$'in cezalandırılması sağlanılır. Bazı özel kısıtlar içinse bilinen ceza terimleri vardır [14] ve bu terimler doğrudan amaç fonksiyonuna eklenebilir. Gezgin satıcı problemi ve varyantları [21-22], çizelgeleme problemleri [23-24], müzik besteleme [25], doğrulanabilirlik problemi [26] gibi pek çok problem için QUBO formülasyonları önerilmiştir. QUBO probleminin çözümü için benzetilmiş tavlama [27], karınca kolonisi algoritması [28] gibi klasik algoritmaların yanı sıra, kuantum tavlama [29], kuantum yaklaşık optimizasyon algoritması [1] gibi kuantum algoritmalar kullanılabilir.

2.3. Ising Modeli

n tane parçacığın her birinin spin adı verilen -1 ya da 1 durumlarından birinde olduğunu durumda, parçacıkların alabileceği 2^n durumdan her birine bir spin konfigürasyonu denir. Ising modeli, spin konfigürasyonlarının özelliklerini analiz etmek için kullanılan bir matematiksel modeldir. Parçacıklar arasındaki etkileşim gücü veya bağlantı kuvveti J_{ij} ile ve her bir parçacığa uygulanan dış kuvvet h_i ile gösterilir. Bir konfigürasyonun enerjisi şu şekilde ifade edilir:

$$\sum_{i<j} J_{ij} s_i s_j + \sum_i h_i s_i \quad \textbf{(5)}$$

Burada $s_i \in \{-1,1\}$ olmak üzere spin değişkenidir. En düşük enerjiye sahip konfigürasyonun bulunması bir minimizasyon problemidir.

Verilen bir QUBO modeli, $x_i = \frac{1-s_i}{2}$ formülüyle denk bir Ising modele dönüştürülebilir. Böylece, bu iki modelin birbirine denk olduğu görülür.

Ising formülasyonuyla ifade edilen bir problem, aynı zamanda aşağıda verilen Hamilton işlemcisinin en düşük enerjili durumuna denk gelmektedir. Burada Z_i, i kübiti üzerine etki eden Pauli-Z operatörüdür. $\psi \in \{0,1\}$ için $Z|\psi\rangle = -1^{\psi}|\psi\rangle$ olduğundan, Eşitlik (6) ile ifade edilen Hamilton işlemcisinin enerjisinin Eşitlik (5) ile ifade edilebileceği görülebilir.

$$H = \sum_{i<j} J_{ij} Z_i Z_j + \sum_i h_i Z_i \quad \textbf{(6)}$$

Sonuç olarak, 0-1 programlama formülasyonuyla verilen bir kombinatorik optimizasyon problemi, önce QUBO'ya, ardından Ising modele çevrilebilir. Ising modele çevrildikten sonra, en düşük enerjili durumu bu Ising modele denk gelen bir Hamilton işlemcisi yazılabilir. Böylece, çözülmek istenen problem, Hamilton işlemcisinin en düşük enerjili konfigürasyonunu bulma problemine indirgenmiş olur. Bu problem, genel bir Hamilton işlemcisi için NP-Zor bir problemdir.

Daha genel formuyla, herhangi bir kombinatorik optimizasyon problemi, n bit ikili x dizgileri üzerinde $f(x)$: $\{0,1\}^n \rightarrow R$ şeklinde tanımlanabilir. $f(x)$'e karşılık gelen ve $H|\psi\rangle = f(x)|\psi\rangle$ şartını sağlayan bir H Hamilton işlemcisi, [16] kaynağında verilen yöntemlerle bulunabilir.

2.4. QAOA

Kuantum Yaklaşık Optimizasyon Algoritması (Quantum Approximate Optimization Algorithm - QAOA) Farhi vd. tarafından tasarlanmış kombinatorik optimizasyon problemlerinin yaklaşık çözümünü bulmak için tasarlanmış, klasik-kuantum hibrit bir algoritmadır [1]. QAOA devresi aşağıdaki şekilde ifade edilebilir.

$$\begin{aligned} |\gamma,\beta\rangle &= \prod_{k=1}^{p} exp(-i\beta_k H_M) exp(-i\gamma_k H_C)|+\rangle \\ &= exp(-i\beta_p H_M) exp(-i\gamma_p H_C) \cdots \\ &\quad exp(-i\beta_1 H_M) exp(-i\gamma_1 H_C)|+\rangle \end{aligned} \quad \textbf{(7)}$$

Burada, p katman sayısı, $|+\rangle$ eşit süperpozisyon durumu, $H_M = -\sum_i X_i$ karıştırıcı Hamilton işlemcisi, H_C köşegen amaç fonksiyon Hamilton işlemcisidir. γ ve β klasik olarak optimize edilmesi gereken parametrelerdir. p katman için $2p$ parametre vardır. Hesaplama bazında yapılan ölçümler sonucunda, H_C'nin beklenen değeri

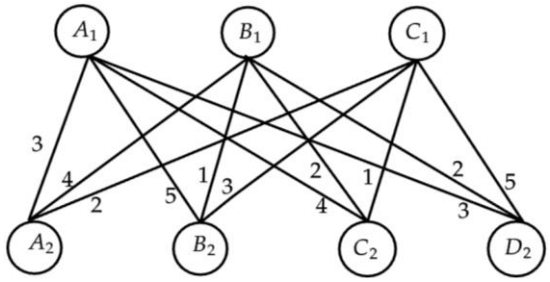
$$\langle H_C \rangle = \langle \gamma,\beta | H_C | \gamma,\beta \rangle \quad \textbf{(8)}$$

şeklinde hesaplanabilir. Bu hesaplama tekrarlanarak, klasik bir optimizasyon yöntemiyle $\langle H_C \rangle'yi$ minimize etmek üzere bir sonraki iterasyon için yeni parametreler elde edilir. Böylece, QAOA algoritması ile $\langle H_C \rangle$ minimize edilmiş olur.

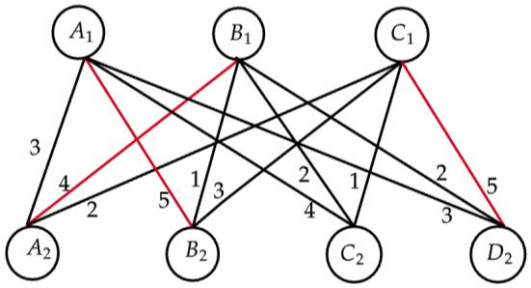
QAOA algoritması, max-cut [30], gezgin satıcı problemi [31], sırt çantası [32], gibi pek çok problem için uygulanmıştır. Ayrıca, literatürde pek çok farklı varyantı önerilmiştir [4, 33-34].

III. PROBLEM TANIMI VE MATEMATİKSEL FORMÜLASYON

Bu bölümde iki parçalı ağırlıklı tam graf üzerinde eşleştirme probleminin matematiksel tanımını yapacak, ardından 0-1 programlama formülasyonunu verecek ve ceza ve Boolean temsil yöntemleri ile Hamilton işlemcisi inşasını göstereceğiz.



Şekil 1. İki parçalı ağırlıklı tam graf örneği. Bu örnekte $V_1 = \{A_1, B_1, C_1\}$ ve $V_2 = \{A_2, B_2, C_2, D_2\}$ olmak üzere 7 düğüm ve 12 kenar bulunmaktadır. Kenar ağırlıkları kenarların yanında belirtilmiştir.



Şekil 2. Maksimum eşleşmede yer alan kenarlar kırmızı ile gösterilmiştir. Eşleşmenin toplam ağırlığı 14'tür.

3.1. İki parçalı ağırlıklı tam graf üzerinde eşleştirme problemi

Bir graf iki parçalı graf ise, aynı kümedeki düğümler arasında hiç kenar olmayacak şekilde düğümler iki kümeye ayrılabilir. Kenarlar, yalnızca iki küme arasında mevcuttur. Olası tüm kenarların yer aldığı durumda, bu grafa iki parçalı tam graf denir. Kenarların üzerinde ağırlık olduğu durumda, iki parçalı ağırlıklı tam graf elde etmiş oluruz.

Matematiksel olarak, iki parçalı tam graf, $V_1 \cup V_2$ düğüm kümesi, E kenar kümesi, $|V_1| = n_1$ ve $|V_2| = n_2$ olmak üzere $K_{n_1,n_2} = (V_1, V_2, E)$ şeklinde tanımlanır. Bu graf üzerinde yer alan kenarlar, $(v_1, v_2) \in E$ eğer $v_1 \in V_1$ ve $v_2 \in V_2$ ise şeklinde ifade edilebilir. Her kenarın bir ağırlığa sahip olduğu durumda, $W: E \to R$ şeklinde her kenara bir ağırlık atanacaktır. Örnek bir graf, Şekil 1'de verilmiştir. Maksimum eşleştirme probleminde amaç E'nin şu şartları sağlayan bir alt kümesini seçmektir: Seçilen kenarların ağırlıkların toplamı maksimize edilmeli, seçilen kenarların birbirleriyle ortak düğümü olmamalıdır. Şekil 1'de verilen graf için maksimum eşleşme Şekil 2'de gösterilmiştir.

Tüm ağırlıkların pozitif olduğunu varsayalım. $n_1 = n_2 = n$ durumunda, maksimum eşleşmede n kenar bulunur. Bu tarz eşleşmelere mükemmel eşleşme denir. $n = n_1 < n_2$ ve $n = n_2 < n_1$durumlarında da maksimum eşleşmede n kenar bulunur.

$n_1 = n_2 = n$ durumuna geri dönersek, tüm ağırlıkların pozitif olduğu durumda, olası tüm eşleşmelerin sayısı $n!$ olacaktır. Bu durumda, problem en büyük ağırlıklı permütasyonu bulma problemine dönüşür.

Bazı ağırlıkların negatif olduğu, yani, bazı düğüm çiftlerinin hiçbir koşulda eşleştirilmesinin istenmediği durumlarda, maksimum eşleşmede n kenar bulunmak zorunda değildir. Bu durumda olası eşleşme sayısını şu şekilde bulabiliriz: Eşleşmede m kenar olduğu durumda, V_1kümesinden n düğüm içinden m tanesini $\frac{n}{m}$ farklı şekilde seçeriz; V_2 kümesinden n düğüm içinden m tanesini $\frac{n}{m}$ farklı şekilde seçeriz; . m düğüm, m düğümle $m!$ farklı şekilde eşleşebilir. Bu durumda m kenar için olası $\frac{n^2}{m} m!$ seçenek vardır. $\sum_{m=0}^{n} \frac{n^2}{m} m!$ olası farklı eşleştirme yapılabilir. Örneğin, n=2 durumu için, 0 kenarın yer aldığı 1 eşleşme, 1 kenarın yer aldığı 4 farklı eşleşme, 2 kenarın yera aldığı 2 farklı eşleşme, toplamda 7 farklı eşleşme vardır. Tablo 1'de, farklı n değerleri için eşleşme sayıları verilmiştir.

Tablo 1. Farklı n değerleri için eşleştirme sayıları

n	Kenar sayısı	Eşleştirme sayısı
2	4	7
3	9	34
4	16	209
5	25	1546
6	36	13327

3.2. 0-1 Programlama Formülasyonu

İki parçalı ağırlıklı tam graf üzeride maksimum eşleştirme problemi, 0-1 programlama formülasyonu kullanılarak aşağıdaki şekilde ifade edilebilir:

$$min \; -\sum_{i \in V_1} \sum_{j \in V_2} w_{i,j} x_{i,j} \quad \textbf{(9)}$$

$$\sum_{j \in V_2} x_{i,j} \leq 1 \quad \forall i \in V_1 \quad \textbf{(10)}$$

$$\sum_{i\in V_1} x_{i,j} \leq 1 \quad \forall j \in V_2 \tag{11}$$

$$x_{i,j} \in \{0,1\} \quad i \in V_1 \; j \in V_2 \tag{12}$$

$x_{i,j}$ karar değişkenleri şu şekilde tanımlanmaktadır:

$$x_{i,j} = 1 \text{ eğer } (i,j) \text{ kenarı eşleştirmede yer alıyorsa} \tag{13}$$

$$0 \text{ aksi takdirde} \tag{14}$$

$x_{i,j}$'nin 1 olduğu durum, (i,j) kenarının eşleştirmede yer aldığı durumu temsil etmektedir. Maaliyet fonksiyonu, $w_{i,j}x_{i,j}$ ifadesini tüm i ve j kombinasyonları için maksimize etmektedir. Minimizasyon şeklinde ifade etmek için Eşitlik (9)'da maaliyet fonksiyonu -1 ile çarpılmıştır. İlk kısıt, V_1kümesindeki düğümlerin V_2 kümesinden en fazla 1 düğümle eşleşebileceğini, ikinci kısıt ise V_2kümesindeki düğümlerin V_1 kümesinden en fazla 1 düğümle eşleşebileceğini ifade etmektedir. Bu formülasyonda gereken değişken sayısı $n_1 \cdot n_2$'dir.

$n_1 = n_2$ durumunda, yukarıda verilen eşitsizlik kısıtları eşitlikle güncellenebilir. Çünkü bu durumda eşleşmede tüm düğümler yer alacaktır. Bu durumda elde edilecek kısıtlar gezgin satıcı problemi, kuadratik eşleştirme problemi gibi pek çok problemde yer alan kısıtlara dönüşür. Bu problemler, maaliyet fonksiyonları açısından farklılık gösterir.

Eğer graf üzerinde negatif ağırlıklar varsa, ya da belli bir sınırın altındaki ağırlıklar eşleştirilmek istenmiyorsa, bu durumda bu kenarlar graftan çıkarılabilir. Ya da başlangıç olarak bazı kenarlar graf üzerinde mevcut olmayabilir. Bu durumda $N(j)$, j düğümünün komşuları olmak üzere 0-1 formülasyonu aşağıdaki şekilde ifade edilebilir.

$$min \; -\sum_{(i,j)\in E} w_{i,j}x_{i,j} \tag{15}$$

$$s.t. \quad \sum_{j\in N(i)} x_{i,j} \leq 1 \quad \forall i \in V_1 \cup V_2 \tag{16}$$

$$x_{i,j} \in \{0,1\} \quad (i,j) \in E \tag{17}$$

Böylece değişken sayısı E pozitif ağırlığa sahip kenarlar kümesi olmak üzere $|E| < n_1 \cdot n_2$ olacaktır.

3.3. Ceza Yöntemi ile Hamilton İşlemcisi İnşası

0-1 programlama formülasyonu, bir önceki bölümde belirtildiği üzere bir QUBO formülasyonuna dönüştürülebilir. Formülasyondaki eşitsizlikler $\sum_i x_i \leq 1$ formundadır. Bu tarz eşitsizlikler, amaç fonksiyonuna $\sum_i \sum_{j:i<j} x_i x_j$ terimlerinin eklenmesiyle QUBO formuna dönüştürülebilir [14]. Rosenberg [15] tarafından önerilen yönteme göre avantajı, Rosenberg'in yönteminde gerektiği gibi ek olarak yeni değişkenlere ihtiyaç duyulmamasıdır. Buradan yola çıkarak, QUBO formülasyonu aşağıdaki şekilde ifade edilebilir:

$$-\sum_{i\in V_1}\sum_{j\in V_2} w_{i,j}x_{i,j} + A_1\sum_{i\in V_1}(\sum_{j\in V_2}\sum_{k>j} x_{i,j}x_{i,k}) + A_2\sum_{j\in V_2}(\sum_{i\in V_1}\sum_{k>i} x_{i,j}x_{k,j}) \tag{18}$$

Burada, A_1 ve A_2 pozitif ceza katsayılarıdır. QUBO formülasyonun gerektirdiği değişken sayısı $n_1 \cdot n_2$'dir. Bu QUBO formülasyonuna karşılık gelen Ising Modeli bulmak için $x_{i,j}$ ikili değişkenlerini, $s_{i,j} \in \{-1,1\}$ spin değişkenleri olmak üzere $\frac{1-s_{i,j}}{2}$ ile değiştirmemiz gerekir.

$$-\sum_{i\in V_1}\sum_{j\in V_2} w_{i,j}\frac{1-s_{i,j}}{2} + A_1\sum_{i\in V_1}(\sum_{j\in V_2}\sum_{k>j}\frac{1-s_{i,j}}{2}\frac{1-s_{i,k}}{2}) + A_2\sum_{j\in V_2}(\sum_{i\in V_1}\sum_{k>i}\frac{1-s_{i,j}}{2}\frac{1-s_{k,j}}{2}) - \frac{1}{2}\sum_{i\in V_1}\sum_{j\in V_2} w_{i,j} + \frac{1}{2}\sum_{i\in V_1}\sum_{j\in V_2} s_{i,j} + A_1\frac{1}{4}\sum_{i\in V_1}(\sum_{j\in V_2}\sum_{k>j} 1 - s_{i,j} - s_{i,k} + s_{i,j}s_{i,k}) + \frac{1}{4}A_2\sum_{j\in V_2}(\sum_{i\in V_1}\sum_{k>i} 1 - s_{i,j} - s_{k,j} + s_{i,j}s_{k,j}) \tag{19}$$

Yukarıda anlatıldığı üzere, $s_{i,j}$ spin değişkenlerini $Z_{i,j}$ operatörleriyle değiştirerek problemin Hamilton işlemcisini bulabiliriz. Hamilton işlemcisi Eşitlik (20)'de verilmiştir.

$$H_{QUBO} = -\frac{1}{2}\sum_{i\in V_1}\sum_{j\in V_2} w_{i,j}I + \frac{1}{2}\sum_{i\in V_1}\sum_{j\in V_2} Z_{i,j} + \frac{1}{4}A_1\frac{1}{4}\sum_{i\in V_1}(\sum_{j\in V_2}\sum_{k>j} I - Z_{i,j} - Z_{i,k} + Z_{i,j}Z_{i,k}) + \frac{1}{4}A_2\sum_{j\in V_2}(\sum_{i\in V_1}\sum_{k>i} I - Z_{i,j} - Z_{k,j} + Z_{i,j}Z_{k,j}) \tag{20}$$

3.4. Boolean Temsil Yöntemi ile Hamilton İşlemcisi İnşası

Hamilton işlemcisinin oluşturulması için iki eşitsizlik kısıtı ve maaliyet terimlerine karşılık gelen Hamilton işlemcisinin elde edilmesi gerekir. Hadfield vd. [16] makalesinde, bazı Boolean temsillere karşılık gelen Hamilton işlemcisi terimlerini vermiştir. Bunlardan bazıları aşağıdaki tabloda özetlenmiştir. Bu bölümde, aşağıdaki tablodan faydalanarak Hamilton işlemcilerini inşa edeceğiz.

Tablo 2. Boolean temsiller ve onlara karşılık gelen Hamilton işlemcileri [16].

Boolean temsil	Hamilton işlemcisi
x	$\frac{1}{2}\mathrm{I} - \frac{1}{2}\mathrm{Z}$
$\neg x$	$\frac{1}{2}\mathrm{I} + \frac{1}{2}\mathrm{Z}$
$\wedge_j x_j$	$\frac{1}{2^k}\prod(\mathrm{I} - Z_j)$

3.4.1. Eşitsizlik Kısıtı

0-1 formülasyonunda iki adet eşitsizlik kısıtı vardır ve aynı şekilde modellenecektir. Detaylı analiz açısından ikincisini ele alacağız. Bu kısıt, sabit bir j değeri için $x_{i,j}$ değişkenlerinin en fazla bir tanesinin 1 olduğu duruma karşılık gelmektedir. $j = 0$ için, bunu mantıksal olarak aşağıdaki şekilde ifade edebiliriz:

Tablo 3. eşitsizlik kısıtının sağlandığı durumlarda 1 olan değişken ve bu durumun mantıksal ifadesi. [16].

1 olan değişken	Boolean temsil
$x_{0,j}$	$(x_{0,j} \wedge \neg x_{1,j} \wedge \neg x_{2,j} \wedge \cdots \wedge \neg x_{n_1-1,j})$
$x_{1,j}$	$(\neg x_{0,j} \wedge x_{1,j} \wedge \neg x_{2,j} \wedge \cdots \wedge \neg x_{n_1-1,j})$
$x_{2,j}$	$(\neg x_{0,j} \wedge \neg x_{1,j} \wedge x_{2,j} \wedge \cdots \wedge \neg x_{n_1-1,j})$
$x_{n_1-1,j}$	$(\neg x_{0,j} \wedge \neg x_{1,j} \wedge \neg x_{2,j} \wedge \cdots \wedge x_{n_1-1,j})$
-	$(\neg x_{0,j} \wedge \neg x_{1,j} \wedge \neg x_{2,j} \wedge \cdots \wedge \neg x_{n_1-1,j})$

Tablo 3'ün ilk 4 satırında, sağdaki ifade, ancak ve ancak soldaki ikili değişken 1 diğer değişkenler 0 olduğu durumda 1 değerini alacaktır. Aksi takdirde 0 değerini alacaktır. 5. satırda ise, hiçbir değişkenin 1 olmadığı durum ele alınmıştır. Sağdaki ifade ancak ve ancak bu durumda 1 değerini alacaktır. Boolean temsillere denk gelen Hamilton işlemcisi ise, Tablo 2'den faydalanarak yazılabilir. $x_{i,j}$ literalleri $\frac{1}{2}(I - Z_{i,j})$, $\neg x_{i,j}$ literalleri $\frac{1}{2}(I + Z_{i,j})$ Hamilton işlemcileri ile ifade edilerek Tablo 4'teki Hamilton işlemcileri elde edilir.

Tablo 4. Tablo 3'teki Boolean temsillere karşılık gelen Hamilton işlemcileri [16].

Hamilton işlemcisi
$\frac{1}{2^{n_1}}(I - Z_{0,j})(I + Z_{1,j})(I + Z_{2,j}) \cdots (I + Z_{n_1-1,j})$
$\frac{1}{2^{n_1}}(I + Z_{0,j})(I - Z_{1,j})(I + Z_{2,j}) \cdots (I + Z_{n_1-1,j})$
$\frac{1}{2^{n_1}}(I + Z_{0,j})(I + Z_{1,j})(I - Z_{2,j}) \cdots (I + Z_{n_1-1,j})$
$\frac{1}{2^{n_1}}(I + Z_{0,j})(I + Z_{1,j})(I + Z_{2,j}) \cdots (I - Z_{n_1-1,j})$
$\frac{1}{2^{n_1}}(I + Z_{0,j})(I + Z_{1,j})(I + Z_{2,j}) \cdots (I + Z_{n_1-1,j})$

Tablo 3'ün ilgili satırındaki Boolean temsil 1 değerini aldığında, Hamilton işlemcisi 1; 0 değerini aldığında Hamilton işlemcisi 0 değerini vermektedir. Boolean temsillerden sadece tek bir tanesi 1 değerini alabilir ya da hepsi 0 değerini alır. Tablo 4'ü kullanarak ceza Hamilton işlemcisini Eşitlik (21)'deki gibi ifade edebiliriz:

$$\begin{aligned} H_{\leq 2,j} \;=\; & I - \frac{1}{2^{n_1}}\Big((n_1+1)I + (n_1-1)\sum_{i_1 \in V_1} Z_{i_1,j} \\ & + (n_1-3)\sum_{i_1,i_2 \in V_1 \times V_1} Z_{i_1,j}Z_{i_2,j} \\ & + (n_1-5)\sum_{i_1,i_2,i_3 \in V_1 \times V_1 \times V_1} Z_{i_1,j}Z_{i_2,j}Z_{i_3,j} \\ & \vdots \\ & + (n_1-2k+1)\sum_{i_1,i_2,\ldots,i_k \in {V_1}^k} Z_{i_1,j}Z_{i_2,j}\cdots Z_{i_k,j} \\ & \vdots \\ & + (-n_1+1)\textstyle\sum_{i_1,i_2,\ldots,i_{n_1} \in {V_1}^{n_1}} Z_{i_1,j}Z_{i_2,j}\cdots Z_{n_1,j}\Big) \end{aligned} \quad \textbf{(21)}$$

Bu Hamilton işlemcisi, her j değeri için maaliyet Hamilton işlemcisine eklenecektir. Yani, 2. kısıt için eklenmesi gereken ceza Hamilton işlemcisi $\sum_{j \in V_2} H_{\leq 2,j}$ şeklinde ifade edilir.

2. kısıt için inşa ettiğimiz Hamilton işlemcisini kullanarak 1. kısıt için de benzer şekilde Hamilton işlemcisi elde edebiliriz.

$$\begin{aligned} H_{\leq 1,i} \;=\; & I - \frac{1}{2^{n_2}}\Big((n_2+1)I + (n_2-1)\sum_{j_1 \in V_2} Z_{i,j_1} \\ & + (n_2-3)\sum_{j_1,j \in V_2 \times V_2} Z_{i,j_1}Z_{i,j_2} \\ & + (n_2-5)\sum_{j_1,j_2,j_3 \in V_2 \times V_2 \times V_2} Z_{i,j_1}Z_{i,j_2}Z_{i,j_3} \\ & \vdots \\ & + (n_2-2k+1)\sum_{j_1,j_2,\ldots,j_k \in {V_2}^k} Z_{i,j_1}Z_{i,j_2}\cdots Z_{i,j_k} \\ & \vdots \\ & + (-n_2+1)\textstyle\sum_{j_1,j_2,\ldots,j_k \in {V_2}^{n_1}} Z_{i,j_1}Z_{i,j_2}\cdots Z_{i,j_{n_1}}\Big) \end{aligned} \quad \textbf{(22)}$$

Bu Hamilton işlemcisi, her i değeri için maaliyet Hamilton işlemcisine eklenecektir. Yani, 2. kısıt için eklenmesi gereken ceza Hamilton işlemcisi $\sum_{i \in V_1} H_{\leq 1,i}$ şeklinde ifade edilir.

3.4.2. Maaliyet Hamilton İşlemcisi

Maaliyet fonksiyonuna karşılık gelen ifade $\sum_{i \in V_1}\sum_{j \in V_2} w_{i,j}x_{i,j}$'dir. $x_{i,j} \to \frac{1}{2}(I - Z_{i,j})$ dönüşümü yapılarak, bu fonksiyona karşılık gelen Hamilton işlemcisi aşağıdaki gibi elde edilir:

$$\begin{aligned} H_C \;=\; & \sum_{i \in V_1}\sum_{j \in V_2} w_{i,j}\frac{1}{2}(I - Z_{i,j}) \\ = \; & \frac{1}{2}\Big(\sum_{i \in V_1}\sum_{j \in V_2} w_{i,j}I - w_{0,0}Z_{0,0} - w_{0,1}Z_{0,1} - \cdots \\ & - w_{n-1,n-1}Z_{n-1,n-1}\Big) \end{aligned} \quad \textbf{(23)}$$

3.4.3. Amaç Fonksiyonu Hamilton İşlemcisi

Amaç fonksiyonu Hamilton işlemcisi H_P, yukarıda bulduğumuz Hamilton işlemcilerinin toplamı olarak ifade edilir.

$$H_P = P\sum_{i \in V_1} H_{\leq 1,i} + P\sum_{j \in V_2} H_{\leq 2,j} + H_C \quad \textbf{(24)}$$

$P > max_{i,j}w_{i,j}$ olacak şekilde seçilmelidir.

3.4.4. İyileştirmeler

Yukarıda değinildiği gibi, $n_1 = n_2$ durumunda, eşitsizlik kısıtları eşitlikle güncellenebilir. Hatta, $n_1 \leq n_2$ durumunda da, V_1 kümesindeki tüm düğümlerin V_2 kümesinden bir düğümle eşleşmesi garanti altında olduğundan, 1. kısıt eşitlik kısıtı olarak alınabilir. V_1 kümesinden bir düğümün V_2 kümesinden hiç bir düğümle eşleşmediği senaryolar saf dışı bırakıldığından, çözüm uzayındaki eleman sayısı azalacak, olurlu çözümlerin tüm çözümlere oranı artacaktr. Benzer şekilde, $n_2 \leq n_1$ durumunda da 2.kısıt eşitlikle güncellenebilir.

2. kısıt eşitlik olarak alındığında, nasıl ifade edebileceğimize göz atalım. Eşitlik (21)'in son satırı, tüm değişkenlerin 0 olduğu duruma tekabül etmektedir. Kısıtı eşitlik olarak güncellediğimizde, muhakkak değişkenlerden birinin 1 değerini alması gerektiğinden, bu satırı çıkarmamız gerekir. $H_{=2,j}$'nin genel formundaki terimlerin katsayısını bulmak için, $H_{\leq 2,j}$'de kullandığımız mantığı kullanabiliriz. Farklı olarak, son clause yer almadığından, tüm katsayılar 1 eksik olacaktır. $H_{=2,j}$'nin genel formunu Eşitlik (25)'teki gibi yazabiliriz.

$$
\begin{aligned}
H_{=2,j} \;=\; & I - \frac{1}{2^{n_1}}\Big((n_1)I + (n_1-2)\sum_{i_1 \in V_1} Z_{i_1,j} \\
& + (n_1-4)\sum_{i_1,i_2 \in V_1 \times V_1} Z_{i_1,j}Z_{i_2,j} + \\
& + (n_1-6)\sum_{i_1,i_2,i_3 \in V_1 \times V_1 \times V_1} Z_{i_1,j}Z_{i_2,j}Z_{i_3,j} \\
& \vdots \\
& + (n_1-2k)\sum_{i_1,i_2,\ldots,i_k \in {V_1}^k} Z_{i_1,j}Z_{i_2,j}\cdots Z_{i_k j} \\
& \vdots \\
& + (-n_1)\textstyle\sum_{i_1,i_2,\ldots,i_{n_1} \in V_1^{n_1}} Z_{i_1,j}Z_{i_2,j}\cdots Z_{n_1,j}\Big)
\end{aligned}
\tag{25}
$$

IV. SONUÇLAR VE TARTIŞMA

Bölüm 3.3 ve 3.4'te tartışılan ceza yöntemi ve Boolean temsil yöntemi ile Hamilton işlemcisi inşasının QAOA algoritmasına etkisini anlamak amacıyla her iki Hamilton işlemcisi kullanılarak QAOA algoritmasıyla simülasyonlar gerçekleştirildi. Bu simülasyonlarda Qiskit 1.4.2 ve Python 3.12 sürümleri kullanıldı. Simülasyonlar gürültüsüz olarak 10000 shots kullanılarak gerçekleştirildi ve 500 kez tekrar edilerek her iterasyonun en iyi sonucu alındı. Optimizasyon algoritmaları için durdurma ölçütü 1e-13 seçildi.

Veri seti olarak iki parçalı graflarda mümkün olan üç farklı düğüm kümesi senaryosu dikkate alındı. Bu senaryolar parça grafların eleman sayılarının birbirinden büyük, küçük ya da birbirine eşit olduğu durumlara denk gelecek şekilde seçildi. Bu senaryolarla yapılan simülasyonlarda tohum sırasıyla 63, 30 ve 16 olarak seçildi.

Seçilen düğüm sayıları ve bu şekilde kurulan graflardaki eşleşme sayıları Tablo 5'te verilmektedir. Bu senaryolar için ağırlık matrisleri,

$$
\omega_1 = \begin{pmatrix} 55.84 & 40.23 & 5.76 \\ 53.04 & 16.15 & 50.50 \\ 26.35 & 54.77 & 55.28 \end{pmatrix},
$$

$$
\omega_2 = \begin{pmatrix} 64.77 & 38.69 & 66.64 & 17.20 \\ 96.30 & 35.32 & 99.18 & 24.27 \\ 58.98 & 41.28 & 14.49 & 54.87 \end{pmatrix},
$$

$$
\omega_3 = \begin{pmatrix} 23.11 & 52.79 & 55.52 \\ 5.51 & 36.71 & 23.09 \\ 69.18 & 17.21 & 7.96 \\ 94.16 & 56.80 & 8.72 \end{pmatrix},
$$

olarak rastgele şekilde üretilmiştir.

Tablo 5. Çalışmada ele alınan farklı senaryolar için iki parçalı grafı oluşturan düğüm kümelerinin eleman sayıları arasındaki ilişkiler.

Senaryolar	Düğüm sayısı ilişkileri	n_1	n_2	Eşleşme sayısı
Senaryo 1	$n_1 = n_2$	3	3	34
Senaryo 2	$n_1 < n_2$	3	4	73
Senaryo 3	$n_1 > n_2$	4	3	73

Hadfield'ın yöntemi için Bölüm 3.4.4'te bahsedilen iyileştirme kullanılarak, uygun durumlarda eşitsizlik kısıtları eşitlikle güncellendi. Bu durumda, üç farklı senaryo ve iki farklı yöntem için k-lokal terim sayıları Tablo 6'da verilmiştir. k-lokal terim, k tane $Z_{i,j}$ çarpımından oluşan terim anlamına gelmektedir. Bu terim sayıları Eşitlik 25'te yer alan ifadenin açılımındaki terimler sayılarak bulunmuştur.

Tablo 6'ya bakıldığında Hadfield'ın yönteminin 1 ve 2 lokal terimlere ek olarak yüksek mertebeden terimler ürettiği gözlemlenebilir. Ancak, toplam terim sayısı yanı toplam parametrize kapı sayısı bakımından Boolean temsil yöntemi daha avantajlıdır.

İlk olarak simülasyonda kullanılacak katman sayısını belirlemek için, Boolean temsil yöntemi kullanılarak bütün senaryolarda farklı katman sayıları ile hesaplamalar çalıştırıldı. Uygun katman sayısı belirlendikten sonra, literatürde sık kullanılan optimizasyon metodları dikkate alınarak ceza ve Boolean temsil yöntemlerinin kıyaslaması yapıldı. Bu bölümün geri kalanında, elde edilen sonuçlar tartışılmıştır.

Tablo 6. Boolean temsil ve ceza metoduyla elde edilen modeller için *k*-lokal terim sayıları. PK parametrize kapı sayısını temsil etmektedir ve toplam terim sayısına eşittir.

Boolean Temsil					
n_1, n_2	1-lokal	2-lokal	3-lokal	4-lokal	PK
$n_1 = n_2$	8	9	3	-	20
$n_1 < n_2$	12	0	16	3	31
$n_1 > n_2$	12	0	16	3	31

Ceza			
n_1, n_2	1-lokal	2-lokal	PK
$n_1 = n_2$	9	18	27
$n_1 < n_2$	12	30	42
$n_1 > n_2$	12	30	42

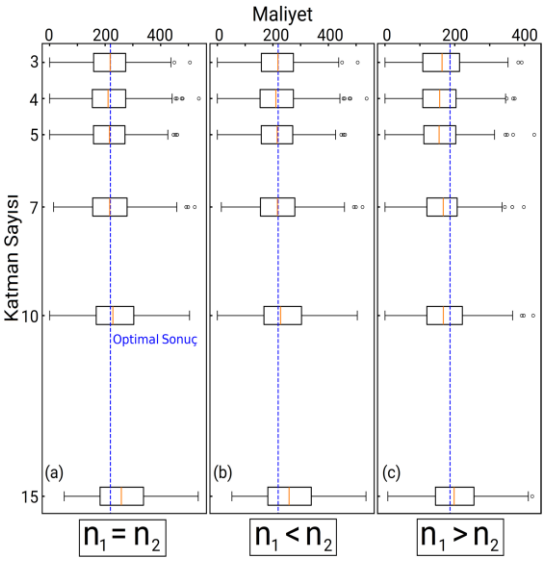
4.1. Katman sayısı etkileri

Katman sayısı arttıkça, QAOA algoritmasının optimal sonuca daha iyi yakınsadığı bilinmektedir. Ancak bu varsayım, parametreler optimal olduğu durumda geçerlidir. Parametrelerin optimizasyonu klasik yöntemlerle yapılmaktadır ve parametre optimizasyonu QAOA algoritması açısından bir darboğaz oluşturmaktadır. Katman sayısı arttıkça parametrelerin sayısı arttığından, parametre optimizasyonu güçleşmektedir. Bu noktada, katman sayısının doğru bir şekilde belirlenmesi önem taşımaktadır.

Katman sayısının sonuçlar üzerinde anlamlı bir etkisi olup olmadığını saptamak adına p = 3, 4, 5, 7, 10, 15 katman sayıları için Boolean temsil yöntemiyle inşa edilen Hamilton işlemcisi kullanılarak QAOA algoritması çalıştırılmıştır. Bu simülasyonlarda COBYLA optimizasyon algoritması kullanılmıştır. Elde edilen maliyetler üzerinde uygulanan tek yönlü ANOVA testi sonucunda $n_1 = n_2$, $n_1 < n_2$ ve $n_1 > n_2$ senaryoları için p değeri sırasıyla 0.0016, 8.01e-22 ve 1.11e-10 olarak bulunmuştur. Böylece, istatiksel olarak katman sayıları arasında anlamlı fark olduğu görülmüştür.

Ele alınan bütün senaryolar için katman sayısı etkisini incelemek üzere yapılmış olan hesaplamaların sonuçları Şekil 3'te verilmektedir. Görüldüğü gibi seçilen en düşük katman sayısında dahi Boolean temsil metodu ile elde edilen sonuçlar optimal sonuç ile tutarlıdır. Tüm senaryolarda, p=3 katman sayısı için yaklaşık çözüm farkı 0'dır, yani optimal çözüm elde edilmiştir. Böylece, Boolean temsil yönteminin, az kuantum kapısı kullanılması tercih edilen NISQ evresi kuantum bilgisayarlarında çalıştırılmaya uygun bir algoritma adayı olduğu söylenebilir.

Tablo 7'de, farklı senaryolarda p=3 ve p=15 durumları için olurlu çözüm yüzdesi, optimal çözüm yüzdesi, ortalama maliyet ve olurlu çözümlerin ortalama maliyeti verilmiştir. 15 katmanlı devrede olurlu ve. optimal çözüm yüzdelerinin 3 katmanlı devreye göre daha büyük olduğu görülmüştür. 15 katmanlı devrede ortalama maliyet 3 katmanlı devreye göre daha büyük olmakla birlikte, Senaryo-2 ve Senaryo-3'te optimal çözümden daha büyüktür. Bu da olurlu olmayan çözümlerin yüksek maliyete sebep olmasından kaynaklanmaktadır.



Şekil 3. Katman sayısının (a) Senaryo-1 (n1 = n2), (b) Senaryo-2 (n1 < n2) ve (c) Senaryo-3 (n1 > n2) için Boolean temsil metodu ile elde edilen sonuçlar üzerindeki etkisi.

Bu durum, zaten eşleşme sayısı yüksek olan bu iki senaryo için, katman sayısının artması sonucu devredeki parametrize kapı sayısının artmasının sonuçları olumsuz etkilemesi olarak yorumlanabilir. Ayrıca, düşük katman sayısı daha az sayıda optimizasyon parametresi anlamına gelmektedir. Tüm senaryolar için 3 katman kullanıldığında optimal sonuca ulaşılabildiğinden, analizin geri kalanında 3 katman kullanılmıştır.

Tablo 7. Ele alınan senaryolarda p=3 ve p=15 katman sayıları için olurlu çözüm yüzdesi, optimal çözüm yüzdesi, ortalama maliyet, olurlu çözümlerin ortalama maliyeti. Optimal maliyet tüm senaryolar için sırasıyla 161,11, 218,82, 186,39'dur.

p	Senaryolar	Olurlu çözüm yüzdesi	Optimal çözüm yüzdesi	Ortalama maliyet	Olurlu çözümlerin ortalama maliyeti
3	$n_1 = n_2$	9,6	1	142,08	113,73
	$n_1 < n_2$	5,4	2	218,19	166,04
	$n_1 > n_2$	2,6	0,6	163,07	124,25
15	$n_1 = n_2$	21	6,6	137,00	139,43
	$n_1 < n_2$	6	1	263,02	174,79
	$n_1 > n_2$	3,2	0,8	188,89	150,59

4.2. Farklı optimizatör etkileri

Bu bölümde ceza ve Boolean temsil yöntemleri için 3 farklı optimizasyon modeli ile elde edilen sonuçları tartışacağız. Literatürde en sık kullanılan optimizasyon modelleri arasından kuantum devrelerinin optimizasyonunda en başarılı adaylar olan SLSQP, COBYLA ve Powell modellerini dikkate aldık [41]. Öncelikle üç farklı senaryo için oluşturduğumuz grafları hem ceza hem de Boolean temsil yöntemleri ile parametrize kuantum devrelerine çevirdik. Bu kısımda QAOA katman sayısını 3 olarak belirledik. Ardından, bu kuantum devrelerini rastgele başlangıç parametreleriyle başlatarak üç optimizasyon modeliyle optimize ettik. Elde edilen sonuçlar Ek'te yer alan Şekil 1-3'te verilmektedir.

Eklerde sunulan bu grafiklere bakıldığında, hem ceza hem de Boolean temsil yöntemleri ile üretilmiş kuantum devrelerinde en fazla sayıda olurlu çözüm bulan optimizasyon modeli Powell modelidir. Aynı zamanda, yine bu grafiklerden görüldüğü gibi Powell optimizasyon modeli ile elde edilen sonuçlar daha düşük bir yayılma sergilemektedir ve bu durum Powell optimizasyon modelinin daha stabil sonuçlar ürettiğini göstermektedir. Bununla birlikte, çalıştırdığımız hesaplarda Powell modelinin, üç optimizasyon modeli arasında en yüksek çalışma süresine sahip olduğu gözlenmiştir. Tüm metodlar, optimal çözüme ulaşabilmiştir.

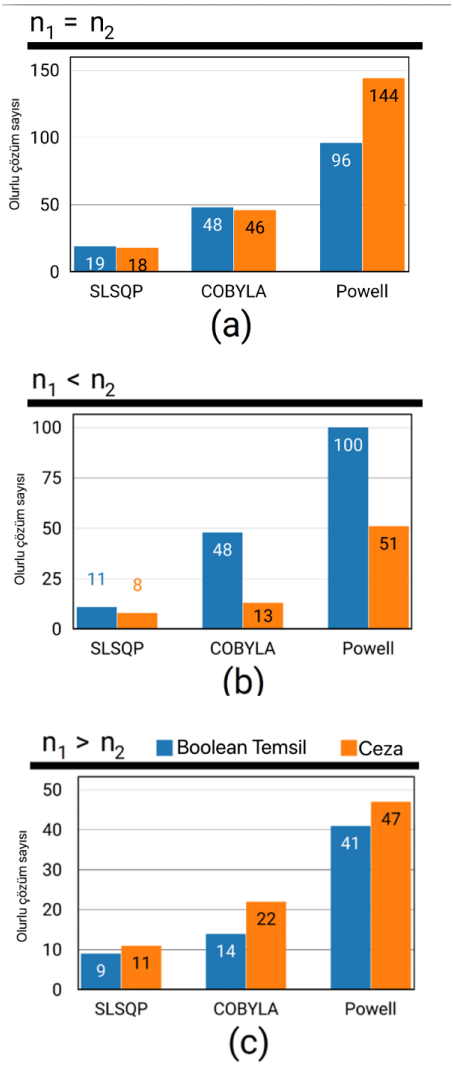
4.3. Simülasyon Sonuçları

Şekil 4'te de görüldüğü üzere, ilk ($n_1 = n_2$) ve üçüncü ($n_1 > n_2$) senaryolar ele alındığında Boolean temsil yönteminin ceza yöntemi ile neredeyse eşit sayıda optimal sonuç üretmiştir. Ancak Boolean temsil yönteminin başarısı Senaryo-2'de ($n_1 < n_2$) ortaya çıkmaktadır. Bu senaryoda Boolean temsil yöntemi özellikle COBYLA ve Powell modelleri ile optimize edildiğinde, ceza yöntemine kıyasla oldukça yüksek sayıda optimal sonuç üretebilmiştir. Elde edilen bu sonuçlardan yola çıkarak, Boolean temsil yönteminin düğüm sayısı farklı olan ve ikinci senaryoya ($n_1 > n_2$) göre düzenlenen iki parçalı graflar için sonuçlar üzerinde bariz bir pozitif etki yarattığı söylenebilir. Neyse ki düğüm sayısı farklı grafları üçüncü senaryoya göre değil ikinci senaryoya göre düzenlemek her zaman mümkündür.

V. SONUÇ

Sonuç olarak, bu makalede, iki parçalı ağırlıklı graf üzerinde eşleştirme problemi için Hamilton işlemcisi inşasında ceza yöntemi ve Hadfield'ın Boolean temsil yöntemi, karşılaştırılmıştır. İki yöntemle elde edilen Hamilton işlemcileri kullanılarak QAOA algoritması uygulanmış ve bu süreçte elde edilen sonuçlar, optimal sonuca yakınlık ve fizibilite açısından incelenmiştir. Elde edilen sonuçlar, her iki yöntemin QAOA algoritması üzerindeki etkilerini net bir şekilde ortaya koymuştur. Boolean temsil yöntemi ile inşa edilen Hamilton işlemcisi kullanıldığında, QAOA'nın optimal sonuca yakınlık açısından daha iyi performans gösterdiği gözlemlenmiştir.

Boolean temsil yöntemi ile iki parçalı ağırlıklı graf eşleşme probleminin optimizasyonunda daha az sayıda parametre ile iyi ve güvenilir sonuçlar elde edilmesi bu yöntemin diğer optimizasyon problemlerinde de önemli sonuçlar verebileceğine işaret etmektedir. Bu nedenle, bu çalışmada sunmakta olduğumuz Boolean temsil Hamilton işlemcisi inşa yöntemi ile diğer optimizasyon problemlerinin ele alınması umut vadedici sonuçlar doğurabilir. Buradan yola çıkarak, benzer kısıt yapısında sahip diğer problemler için de, Boolean temsil yöntemi ile Hamilton işlemcisi inşasının daha verimli olup olmayacağı, daha fazla örnek ve deneysel verilerle desteklenmelidir.



Şekil 4. Boolean temsil ve ceza yöntemlerinin (a) Senaryo-1 ($n_1 = n_2$), (b) Senaryo-2 ($n_1 < n_2$) ve (c) Senaryo-3 ($n_1 > n_2$) için farklı optimizasyon modelleri üzerinden karşılaştırılması. y ekseninde olurlu çözüm sayıları verilmiştir.

Gelecekte, bu yöntemin daha fazla problem sınıfına uygulanması, ve onun getirdiği avantajların daha net bir şekilde ortaya konması önemlidir. Bu çalışmanın ileri aşaması olarak, iki yöntem farklı optimizasyon problemleri üzerinde kıyaslanabilir. Ayrıca, gürültü altında iki metodun nasıl davranacağını incelemek adına, gürültü modelleri kullanılarak simülasyon yapılabilir.

KAYNAKLAR

[1] Farhi, E., Goldstone, J., & Gutmann, S. (2014). A quantum approximate optimization algorithm. arXiv preprint arXiv:1411.4028.

[2] Preskill, J. (2018). Quantum computing in the NISQ era and beyond. Quantum, 2, 79.

[3] Born, M., & Fock, V. (1928). Beweis des adiabatensatzes. Zeitschrift für Physik, 51(3), 167–180.

[4] Hadfield, S., et al. (2019). From the quantum approximate optimization algorithm to a quantum alternating operator ansatz. Algorithms, 12(2), 34.

[5] Golden, J., Bärtschi, A., O'Malley, D., & Eidenbenz, S. (2021). Threshold-based quantum optimization. 2021 IEEE International Conference on Quantum Computing and Engineering (QCE), 137–147.

[6] Sack, S. H., & Serbyn, M. (2021). Quantum annealing initialization of the quantum approximate optimization algorithm. Quantum, 5, 491.

[7] Zhu, L., et al. (2022). Adaptive quantum approximate optimization algorithm for solving combinatorial problems on a quantum computer. Physical Review Research, 4(3), 033029.

[8] Bakó, B., Glos, A., Salehi, Ö., & Zimborás, Z. (2025). Prog-QAOA: Framework for resource-efficient quantum optimization through classical programs. Quantum, 9, 1663.

[9] Hastings, M. B. (2019). Classical and quantum bounded depth approximation algorithms. arXiv preprint arXiv:1905.07047.

[10] Morales, M. E. S., Biamonte, J. D., & Zimborás, Z. (2020). On the universality of the quantum approximate optimization algorithm. Quantum Information Processing, 19, 1–26.

[11] Farhi, E., Gamarnik, D., & Gutmann, S. (2020). The quantum approximate optimization algorithm needs to see the whole graph: A typical case. arXiv preprint arXiv:2004.09002.

[12] Blekos, K., et al. (2024). A review on quantum approximate optimization algorithm and its variants. Physics Reports, 1068, 1–66.

[13] Lucas, A. (2014). Ising formulations of many NP problems. Frontiers in Physics, 2, 5.

[14] Glover, F., Kochenberger, G., & Du, Y. (2018). A tutorial on formulating and using QUBO models. arXiv preprint arXiv:1811.11538.

[15] Rosenberg, I. G. (1975). Reduction of Bivalent Maximization to the Quadratic Case.

[16] Hadfield, S. (2021). On the representation of Boolean and real functions as Hamiltonians for quantum computing. ACM Transactions on Quantum Computing, 2(4), 1–21.

[17] Montgomery, R. J. (2016). Kidney paired donation: Optimal and equitable matchings in bipartite graphs. Rose-Hulman Undergraduate Mathematics Journal, 17(1), 11.

[18] Kesselheim, T., Radke, K., Tönnis, A., & Vöcking, B. (2013). An optimal online algorithm for weighted bipartite matching and extensions to combinatorial auctions. European Symposium on Algorithms, 589–600.

[19] Mehta, A., et al. (2013). Online matching and ad allocation. Foundations and Trends in Theoretical Computer Science, 8(4), 265–368.

[20] Graver, J. E. (1975). On the foundations of linear and integer linear programming I. Mathematical Programming, 9(1), 207–226.

[21] Salehi, Ö., Glos, A., & Miszczak, J. A. (2022). Unconstrained binary models of the travelling salesman problem variants for quantum optimization. Quantum Information Processing, 21(2), 67.

[22] Gonzalez-Bermejo, S., Alonso-Linaje, G., & Atchade-Adelomou, P. (2022). GPS: A new TSP formulation for its generalizations type QUBO. Mathematics, 10(3), 416.

[23] Zhang, J., Bianco, G., & Beck, J. (2022). Solving job-shop scheduling problems with QUBO-based specialized hardware. Proceedings of the International Conference on Automated Planning and Scheduling, 32.

[24] Domino, K., Kundu, A., Salehi, Ö., & Krawiec, K. (2022). Quadratic and higher-order unconstrained binary optimization of railway rescheduling for quantum computing. Quantum Information Processing, 21(9), 337.

[25] Arya, A., Botelho, L., Cañete, F., Kapadia, D., & Salehi, Ö. (2022). Applications of quantum annealing to music theory. In Quantum Computer Music: Foundations, Methods and Advanced Concepts, Springer, 373–406.

[26] Nüßlein, J., Gabor, T., Linnhoff-Popien, C., & Feld, S. (2022). Algorithmic QUBO formulations for k-SAT and hamiltonian cycles. Proceedings of the Genetic and Evolutionary Computation Conference Companion, 2240–2246.

[27] Kirkpatrick, S., Gelatt Jr, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. Science, 220(4598), 671–680.

[28] Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. IEEE Computational Intelligence Magazine, 1(4), 28–39.

[29] Kadowaki, T., & Nishimori, H. (1998). Quantum annealing in the transverse Ising model. Physical Review E, 58(5), 5355.

[30] Crooks, G. E. (2018). Performance of the quantum approximate optimization algorithm on the maximum cut problem. arXiv preprint arXiv:1811.08419.

[31] Glos, A., Krawiec, A., & Zimborás, Z. (2022). Space-efficient binary optimization for variational quantum computing. npj Quantum Information, 8(1), 39.

[32] de la Grand'rive, P. D., & Hullo, J.-F. (2019). Knapsack problem variants of QAOA for battery revenue optimisation. arXiv preprint arXiv:1908.02210.

[33] Bakó, B., Glos, A., Salehi, Ö., & Zimborás, Z. (2022). Prog-QAOA: Framework for resource-efficient quantum optimization through classical programs. arXiv preprint arXiv:2209.03386.

[34] Herrman, R., Lotshaw, P. C., Ostrowski, J., Humble, T. S., & Siopsis, G. (2022). Multi-angle quantum approximate optimization algorithm. Scientific Reports, 12(1), 6781.

[35] Kuhn, H. W. (1955). The Hungarian method for the assignment problem. Naval Research Logistics Quarterly, 2(1–2), 83–97.

[36] Edmonds, J., & Karp, R. M. (1972). Theoretical improvements in algorithmic efficiency for network flow problems. Journal of the ACM (JACM), 19(2), 248–264.

[37] Fredman, M. L., & Tarjan, R. E. (1987). Fibonacci heaps and their uses in improved network optimization algorithms. Journal of the ACM (JACM), 34(3), 596–615.

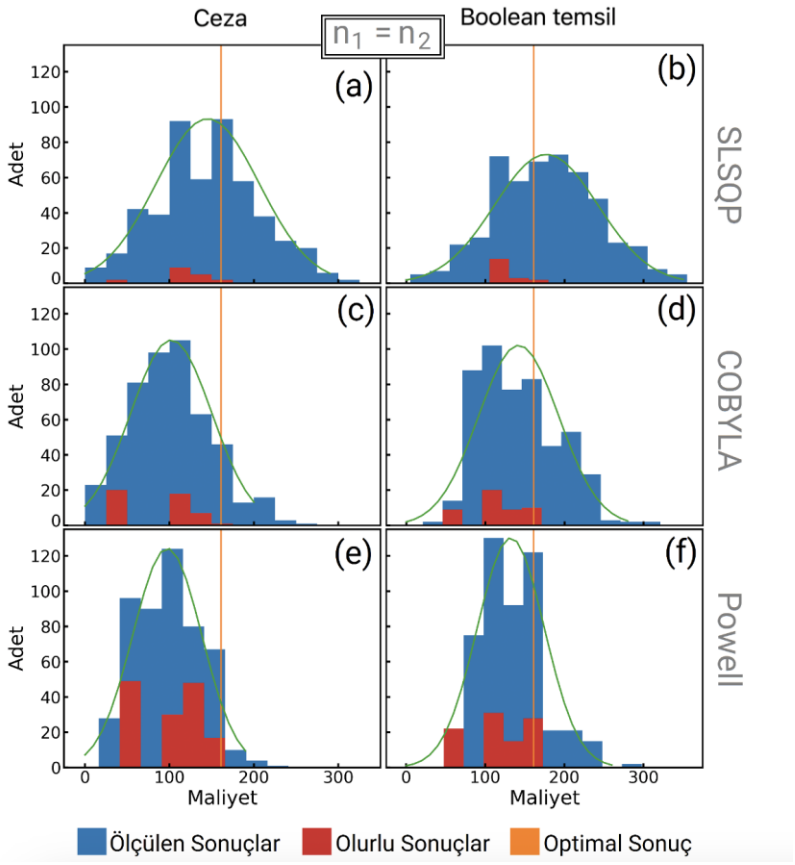
[38] Kaplan, H., Naori, D., & Raz, D. (2022). Online weighted matching with a sample. Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 1247–1272.

[39] Fahrbach, M., Huang, Z., Tao, R., & Zadimoghaddam, M. (2022). Edge-weighted online bipartite matching. Journal of the ACM, 69(6), 1–35.

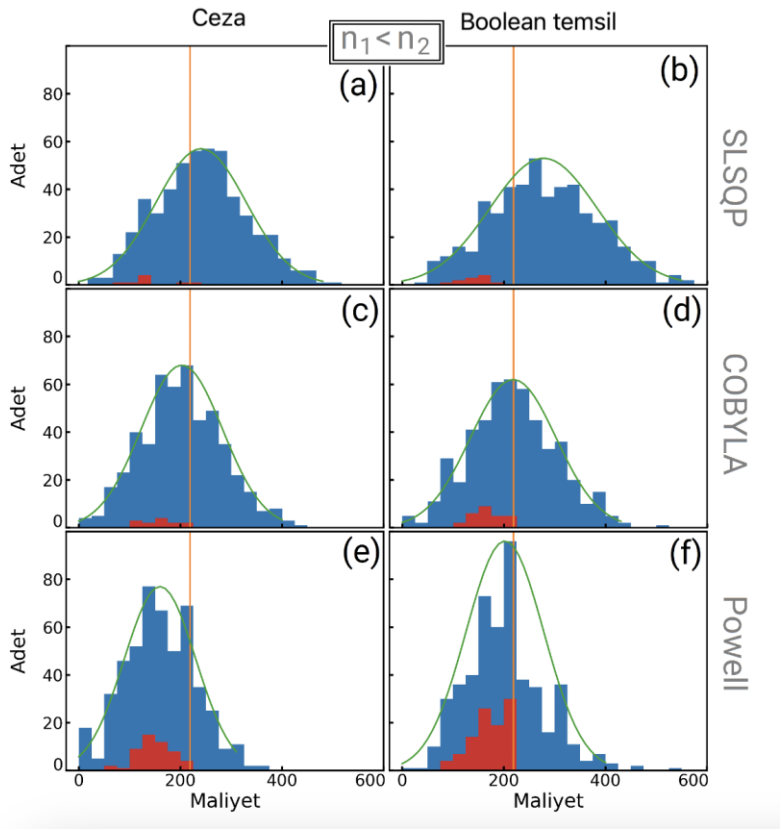
[40] Valencia, C. E., & Vargas, M. C. (2016). Optimum matchings in weighted bipartite graphs. Boletín de la Sociedad Matemática Mexicana, 22, 1–12.

[41] Fernández-Pendás, M., Combarro, E. F., Vallecorsa, S., Ranilla, J., & Rúa, I. F. (2022). A study of the performance of classical minimizers in the quantum approximate optimization algorithm. Journal of Computational and Applied Mathematics, 404, 113388.

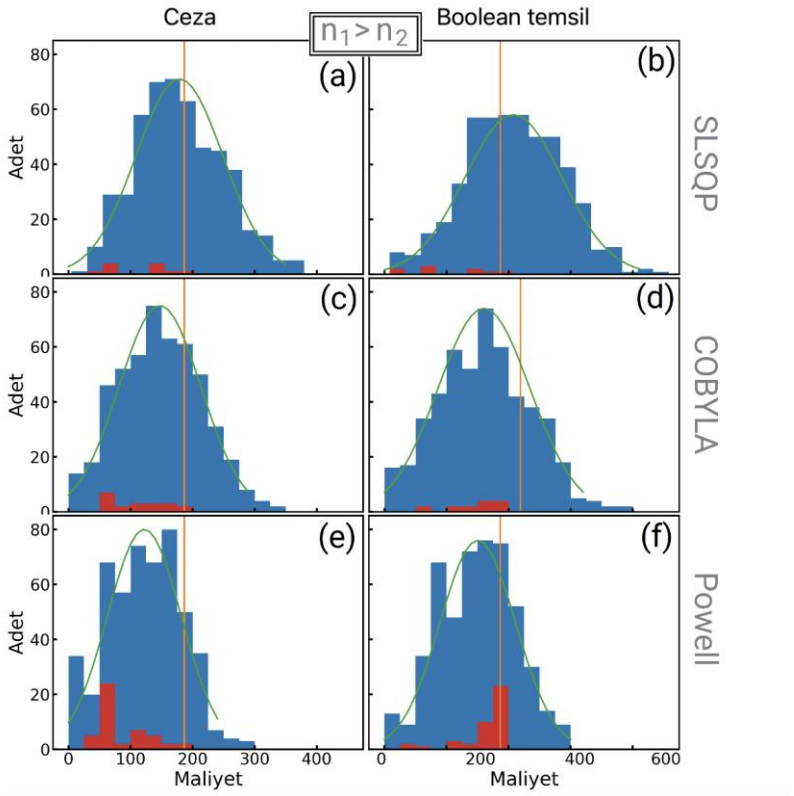
EK



Ek Şekil 1. Senaryo-1 (n_1=n_2) için üç farklı optimizasyon metodu kullanılarak ceza yöntemi ve Boolean temsil metodlarının verdiği sonuçların karşılaştırmalı tablosu. İlk satırda (**(a)** ve **(b)**) SLSQP metodu ile ikinci satırda (**(c)** ve **(d)**) COBYLA metodu ile ve üçüncü satırda (**(e)** ve **(f)**) Powell metodu ile elde edilen optimizasyon grafikleri gösterilirken ilk sütun (**(a), (c)** ve **(e)**) ceza yönteminin ve ikinci sütun (**(b), (d)** ve **(f)**) sonuçlarını içermektedir. Kıyaslamalı tartışma için metne bakınız.



Ek Şekil 2. Senaryo-2 ($n_1<n_2$) için üç farklı optimizasyon metodu kullanılarak ceza yöntemi ve Boolean temsil metodlarının verdiği sonuçların karşılaştırmalı tablosu. İlk satırda (**(a)** ve **(b)**) SLSQP metodu ile ikinci satırda (**(c)** ve **(d)**) COBYLA metodu ile ve üçüncü satırda (**(e)** ve **(f)**) Powell metodu ile elde edilen optimizasyon grafikleri gösterilirken ilk sütun (**(a), (c)** ve **(e)**) ceza yöntemi yönteminin ve ikinci sütun (**(b), (d)** ve **(f)**) sonuçlarını içermektedir. Kıyaslamalı tartışma için metne bakınız.



Ek Şekil 3. Senaryo-3 ($n_1>n_2$) için üç farklı optimizasyon metodu kullanılarak ceza yöntemi ve Boolean temsil metodlarının verdiği sonuçların karşılaştırmalı tablosu. İlk satırda (**(a)** ve **(b)**) SLSQP metodu ile ikinci satırda (**(c)** ve **(d)**) COBYLA metodu ile ve üçüncü satırda (**(e)** ve **(f)**) Powell metodu ile elde edilen optimizasyon grafikleri gösterilirken ilk sütun (**(a), (c)** ve **(e)**) ceza yönteminin ve ikinci sütun (**(b), (d)** ve **(f)**) sonuçlarını içermektedir. Kıyaslamalı tartışma için metne bakınız.