Institute of Architecture of Application Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit

Parameter Initialization for Warm-Starting QAOA

Julian Obst

Course of Study:	Informatik
Examiner:	Prof. Dr. Dr. h. c. Frank Leymann
Supervisor:	Felix Truger, M.Sc., Martin Beisel, M.Sc.
Commenced:	February 15, 2022
Completed:	August 15, 2022

Abstract

The Quantum Approximate Optimization Algorithm (QAOA) is a promising candidate for achieving quantum advantage on near-term quantum devices. Warm-Started QAOA (WS-QAOA) is an improved version of QAOA. It utilizes an approximate solution to the problem in question, which is efficiently computed on classical hardware first. This version aims for improved performance at a lower depth. Since QAOA is a variational algorithm, finding suitable parameters is key to its successful application. Oftentimes, good parameters are obtained by picking random initial parameters and optimizing them. There have been recent works that demonstrated a successful transfer of parameters between different problem instances solved with QAOA. This thesis investigates how parameter transfer can be applied to WS-QAOA for MaxCut with a focus on unweighted 3-, 4- and 5-regular graphs. I show that parameters can be successfully transferred between warm-started d-regular instances, where both instances have degree d. The approximate MaxCut solution used to warm-start the algorithm (initial cut) greatly influences the success and a successful transfer is more likely if both instances are warm-started with a good initial cut. Further, I show empirically that for WS-QAOA the regular instances also demonstrate a concentration of good parameters. For example, in the experiments performed here, optimized parameters of a regular instance could be transferred to another with no significant reduction in approximation ratio on average. The 3- and 5-regular instances even showed an increase of the average approximation ratio by 0.003 and 0.008 respectively, 4-regular instances saw a 0.041 decrease. However, this had more to do with the quality of the initial cut than the transferred parameters. These insights lead to an approach called *landscape approximation* that helps to determine good initial parameters that can be used directly or be further optimized.

Kurzfassung

Der Quantum Approximate Optimization Algorithm (QAOA) ist ein vielversprechender Kandidat für Quantenüberlegenheit auf Near-Term-Hardware. Warm-started QAOA (WS-QAOA) ist eine verbesserte Version von QAOA. Diese nutzt für das vorliegende Problem eine vorberechnete approximative Lösung, welche effizient auf klassischer Hardware bestimmt wird. Diese Version zielt auf bessere Ergebnisse bei geringer Tiefe ab. Da es sich bei QAOA um einen variationellen Algorithmus handelt, ist das Finden von passenden Parametern entscheidend für dessen erfolgreiche Anwendung. Oft werden gute Parameter bestimmt, indem zufällig gewählte initiale Parameter optimiert werden. Aktuelle Arbeiten demonstrierten die erfolgreiche Übertragung von Parametern zwischen verschiedenen Probleminstanzen, die mit QAOA gelöst wurden. Diese Thesis untersucht, wie die Übertragung von Parametern auf WS-QAOA für MaxCut mit einem Fokus auf 3-, 4- und 5-reguläre Graphen angewendet werden kann. Ich zeige, dass eine Übertragung zwischen warm-started d-regulären Instanzen erfolgreich sein kann, wobei beide Instanzen den Grad d haben. Die Näherungslösung des MaxCut-Problems (der sog. initiale Schnitt), welche für den Warm-Start des Algorithmus eingesetzt wird, hat hierbei einen großen Einfluss auf den Erfolg, und eine erfolgreiche Übertragung ist wahrscheinlicher, wenn beide Instanzen einen guten initialen Schnitt haben. Weiter zeige ich empirisch, dass die regulären Instanzen bei WS-QAOA eine Konzentration von guten Parametern aufweisen. Zum Beispiel konnten in den hier durchgeführten Experimenten optimierte Parameter einer regulären Instanz an eine weitere übertragen werden, ohne dass dabei die durchschnittliche Näherungsgüte signifikant reduziert wurde. Die 3- und

5-regulären Instanzen zeigten dabei sogar einen Zuwachs der Güte von 0.003 bzw. 0.008, während die 4-regulären Instanzen eine Reduzierung um 0.041 aufwiesen. Dies hatte mehr mit der Qualität des initialen Schnitts als mit den übertragenen Parametern zu tun. Diese Erkenntnisse führten zu einem Verfahren, genannt *Landscape Approximation*, welches dabei helfen kann, gute initiale Parameter zu bestimmen, die direkt benutzt werden können oder als Initialisierung der Optimierung dienen.

Contents

List of Figures

List of Tables

Acronyms

AQC Adiabatic Quantum Computation. 17

GW Goemans-Williamson. 20

NISQ Noisy Intermediate Scale Quantum. 23

QA Quantum Annealing. 22

QAOA Quantum Approximate Optimization Algortihm. 13

QP Quadratic Programming. 20

QPU Quantum Processing Unit. 13

QUBO Quadratic Unconstrained Binary Optimization. 20

SDP Semidefinite Program. 20

VQA Variational Quantum Algorithm. 13

WS-QAOA Warm-starting QAOA. 14

1 Introduction

At the moment there is a lot of research and investment in the field of quantum computation. Quantum computers exploit quantum mechanical effects to perform computations faster than a classical computer could. A major achievement that demonstrated the potential of quantum computers was in 1994 when Peter Shor discovered an algorithm that could efficiently factor large numbers [23]. To date no known algorithm that this problem with comparable efficiency on a classical computer. Quantum computation promises that at least some problems can be solved more efficiently on quantum devices than any algorithm running on a classical computation device. In practice, however, there are still a lot of hurdles to overcome until quantum computers can be reliably used or surpass their classical counterparts. The reason for this is that the quantum devices available today aren't ideal quantum computers in the sense that calculations are highly susceptible to environmental influences which cause the results to become faulty.

Gate operations aren't totally accurate, each gate introduces slight imperfections so-called gate errors. For example rotations around arbitrary angles are inherently imprecise because some angles can only be approximated. As more and more operations are performed on the Qubits in succession, the more the errors of the calculation accumulate [9]. This is why algorithms that have great properties in theory can't be executed reliably on current quantum devices. As a consequence, there is an increasing interest in algorithms that are robust enough to tolerate a small amount of noise. A promising class of algorithms is the class of the so-called Variational Quantum Algorithms (VQAs) [6]. These are hybrid algorithms, in which a short computation, in form of a parameterized circuit, takes place on a Quantum Processing Unit (QPU) and optimization of these parameters is performed on a classical computer. The loop of execution on a quantum device and optimization on a classical device is repeated until some convergence criterion is reached. When this process is finished, the hope is to be in the possession of the optimal parameters for that circuit. The circuit is finally executed with the optimized parameters to measure a result that is close to the optimum. The idea behind VQAs is that the circuits have a shallow depth and therefore a short computation time. This helps to prevent the accumulation of errors and to get a result that isn't distorted by too much noise.

One example of such an algorithm is the so-called Quantum Approximate Optimization Algortihm (QAOA). It was designed to solve combinatorial optimization problems. The parameterized circuit for this algorithm has two main parts. One is parameterized by an angle γ and corresponds to the problem instance. It is sometimes referred to as the problem circuit. The other part of the circuit is parameterized by β and is called the mixer. At first, some initial state is created, in the most basic case the equal superposition, and problem circuit and mixer are applied in succession. How often the two parts are applied is parameterized by a hyperparameter p. Therefore, p determines the depth of the circuit.

One idea to further improve the performance of QAOA is to lower the depth p of the circuit by changing the initial state of equal superposition. In this case, the quantum circuit's initial state is "warm-started" with a classically precomputed solution. This improvement is called Warm-starting QAOA (WS-QAOA) [10]. Other approaches aim to start with good initial parameters for the optimization loop which could be seen as another form of warm-starting variational algorithms. This is done to save on the number of steps necessary to obtain optimal parameters.

In this thesis I analyze how the parameters γ, β for WS-QAOA can be transferred between 3-, 4- and 5-regular MaxCut instances. This transfer serves as a way to initialize the parameters for further optimization or direct usage. In particular, I investigate the landscapes of the objective function of the different subgraphs. Patterns between the different landscapes explain when and why a transfer between instances is successful. The results are summarized in a map that serves as a tool to identify good donor acceptor pairs. Another result is that the regular instances exhibit a concentration of good parameters when WS-QAOA is applied. This is shown with numerical experiments. From these insights, I give an approach on how to calculate good parameters for small instances that can then be used for larger ones.

2 Background

The QAOA is an algorithm suited for QPUs. Simulating QAOA circuits on classical hardware becomes harder with increasing problem size. There is a point at which the simulation with classical computers becomes infeasible. If there was a quantum computer that could execute such a circuit, this would mean quantum advantage. According to estimates of Dalzell et al. [7] this would be the case when a QAOA circuit contains 420 qubits and 500 clauses. Although there aren't any quantum devices that come close to this limit, QAOA is still a very promising algorithm for the near term.

The following sections give a basic explanation of QAOA and its application to the Maximum Cut problem and WS-QAOA. It is based on the original papers for QAOA and WS-QAOA [10, 11] and explanations of Stęchły [24] and Weidenfeller [28].

2.1 Maximum Cut Problem

The Maximum Cut problem, or MaxCut for short, is a problem in combinatorial optimization. The goal of the MaxCut problem is, given a graph, find a partition of the nodes into two parts (a so-called cut), such that the number of edges between the partitions is maximized. For general graphs, finding the optimal cut is NP-hard [25]. An example of a MaxCut is given in Figure 2.1. The Nodes are either in the red or blue partition, and the edges that contribute to the cut are highlighted in orange. The value of the cut is 7.

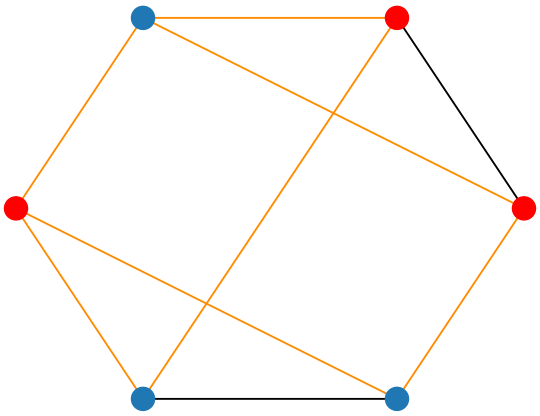

Figure 2.1: MaxCut example

2.2 QAOA

This section gives an explanation of the QAOA algorithm. To distinguish this standard version from WS-QAOA, in this thesis it is sometimes referred to as "cold-started" QAOA. It was first described by Farhi et al. [11] and demonstrated how it can be used to find approximate solutions to the MaxCut problem. In particular, they analyzed the performance on 3-regular graphs, these are graphs where each node has exactly 3 neighbors. To this day, a lot of analysis of QAOA is done with MaxCut as a benchmark.

QAOA aims to approximately solve combinatorial optimization problems. Such problems are defined by n bits and m clauses. A clause is a constraint on the problem and this clause can either be satisfied or not, depending on the assignment of the bits. From this, we can create an objective function.

$$f(z) := C(z) = \sum_{\alpha=1}^{m} C_\alpha(z) \tag{2.1}$$

$z = z_1 z_2 \dots z_n$ is a bit string and $C_\alpha(z) = 1$ if z satisfies clause α and 0 otherwise. In the quantum computer, a bit string z is mapped to an element of the computational basis $|z\rangle \in \mathbb{H}^n$. With this, C induces a hermitian operator $\mathbb{H}^n \to \mathbb{H}^n$, where $C|z\rangle := \sum_{\alpha=1}^{m} C_\alpha(z) \cdot |z\rangle = f(z) \cdot |z\rangle$. In other words, C is a diagonal matrix, whose entries are the values of f.

$$C = \begin{pmatrix} f(0\cdots00) & & & \\ & f(0\cdots01) & & \\ & & \ddots & \\ & & & f(1\cdots11) \end{pmatrix}$$

Notice, that the elements $|z\rangle$ of the computational basis are eigenvectors of C and $f(z)$ is the corresponding eigenvalue. Suppose, we want to maximize $f(z)$. We can do this by computing the maximum eigenvalue of C and with the corresponding eigenvector $|z\rangle$, we have found the desired bit string. Since C is diagonal, we could just read the maximum eigenvalue off of the diagonal.

Now, there is an obvious problem with this procedure. With an increasing number of bits n, the matrix C grows exponentially. Namely, the diagonal has 2^n entries in total. But still, we could find the optimum bit string, by finding the eigenvector corresponding to the optimum eigenvalue.

So what is the procedure of QAOA? First, let's define the parts that are needed to implement QAOA. From the operator C, we define the unitary operator $U(C, \gamma)$ which depends on an angle γ.

$$U(C, \gamma) := e^{-i\gamma C} = \prod_{\alpha=1}^{m} e^{-i\gamma C_\alpha} \tag{2.2}$$

This is the so-called "cost operator" Additionally we define X_j and B as

$$X_j := I \otimes \cdots \otimes I \otimes \underbrace{X}_{j} \otimes I \otimes \cdots \otimes I \tag{2.3}$$

$$B := \sum_{j=1}^{n} X_j \tag{2.4}$$

From this, we construct a similar unitary operator

$$U(B,\beta) := e^{-i\beta B} = \prod_{j=1}^{n} e^{-i\beta X_j} \tag{2.5}$$

the so-called "Mixer". The initial state $|s\rangle$ is the uniform superposition over all computational basis states:

$$|s\rangle = \frac{1}{\sqrt{2^n}} \sum_{z} |z\rangle \tag{2.6}$$

For any integer $p \geq 1$ and $2p$ angles $\gamma_1 \dots \gamma_p \equiv \boldsymbol{\gamma}$ and $\beta_1 \dots \beta_p \equiv \boldsymbol{\beta}$ we construct the quantum state

$$|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle = U(B,\beta_p)U(C,\gamma_p)\cdots U(B,\beta_1)U(C,\gamma_1)\,|s\rangle \tag{2.7}$$

by alternating between the cost operator and the mixer, applied to $|s\rangle$. We can translate this procedure into a circuit. The resulting circuit is depicted in Figure 2.2.

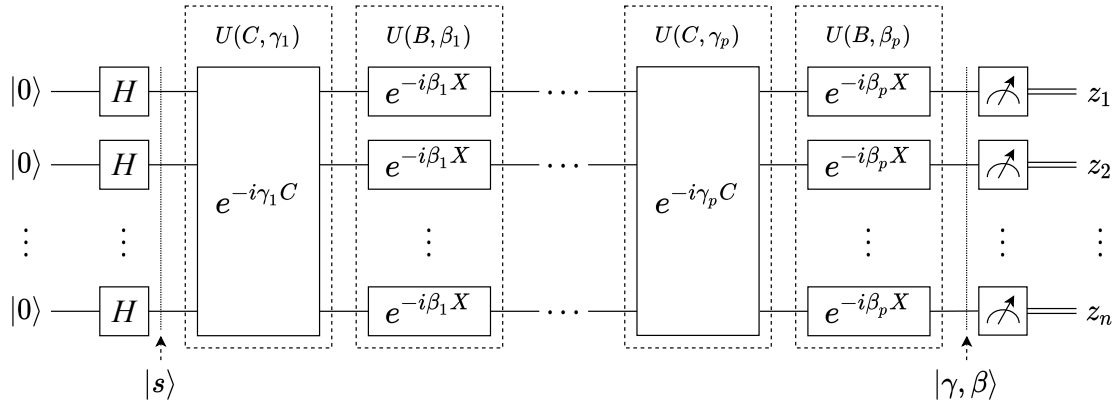

Figure 2.2: QAOA circuit

Measuring the outcome lets us calculate the expectation value $\langle C\rangle_{|\boldsymbol{\gamma},\boldsymbol{\beta}\rangle} = \langle \boldsymbol{\gamma}, \boldsymbol{\beta}|\, C\, |\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle$. For instance, we could run the circuit once to get some bit string z and calculate $C(z)$ on a classical computer. If we do this enough times we can take the mean of these results to get a good approximation for $\langle C\rangle_{|\boldsymbol{\gamma},\boldsymbol{\beta}\rangle}$. In this thesis, the terms expectation value and energy are used interchangeably.

In some cases, the optimum angles can be deterministically computed, as was done in the original QAOA paper [11]. If this is not the case one can try to iteratively optimize the angles $\boldsymbol{\gamma}, \boldsymbol{\beta}$. This is achieved in an optimization loop. Start with some angles $\boldsymbol{\gamma}, \boldsymbol{\beta}$ and execute the aforementioned circuit. Evaluate the circuit to calculate $\langle C\rangle_{|\boldsymbol{\gamma},\boldsymbol{\beta}\rangle}$. Use a classical optimizer to improve the angles $\boldsymbol{\gamma}, \boldsymbol{\beta}$ and to optimize the value of $\langle C\rangle_{|\boldsymbol{\gamma},\boldsymbol{\beta}\rangle}$. Run the circuit with the optimized angles again and repeat this procedure until some stopping criterion is reached.

This is a high-level overview of QAOA. To understand why these steps are taken, we need to understand the following two things: The idea of Adiabatic Quantum Computation (AQC) and Trotterization.

Adiabatic Quantum Computation (AQC)

The Schrödinger equation describes what happens if we evolve a quantum system described by a time-independent hamiltonian H for some time t.

$$i\hbar\frac{d}{dt}|\psi(t)\rangle = H|\psi(t)\rangle \tag{2.8}$$

Solving this equation lets us know the state of the system at some time t if we know the initial state $|\psi(0)\rangle$.

$$|\psi(t)\rangle = e^{-iHt}|\psi(0)\rangle \tag{2.9}$$

AQC makes use of the Adiabatic Theorem [3]. It states, that if we evolve a quantum system with a time-dependent Hamiltonian $H(t)$ with $t \in [0, 1]$ and we start in a ground state of $H(0)$: $|\psi(0)\rangle$, which is the eigenstate corresponding to the lowest eigenvalue, and change t sufficiently slowly, then the quantum system will end up in the ground state of $H(1)$: $|\psi(1)\rangle$. The same holds for the highest energy eigenstate. For the purpose of QAOA, this means we can find the lowest/highest energy eigenstate of a hamiltonian H_C, by starting with a known ground state/highest energy eigenstate of a simple hamiltonian H_B and slowly change it to resemble the hamiltonian H_C. A time-dependent hamiltonian that would achieve this would be

$$H(t) = (1-t)H_B + tH_C \tag{2.10}$$

Since this hamiltonian is not time-independent, we can't use Equation (2.9) to obtain the final state.

Trotterization

Suppose we have a hamiltonian $H = H_1 + H_2$. In general for commuting matrices A, B it holds that

$$e^{A+B} = e^A e^B \tag{2.11}$$

If H_1 and H_2 commute and are time-independent, we could use this fact for the time evolution.

$$e^{-i(H_1+H_2)t} = e^{-iH_1t}e^{-iH_2t} \tag{2.12}$$

The hamiltonian of Equation (2.10) introduced with AQC is neither time-independent nor do H_B and H_C commute. However, the evolution can be approximated with the Trotter-Suzuki-Formula.

$$e^{-i(H_1+H_2)t} \approx (e^{-iH_1t/r}e^{-iH_2t/r})^r \tag{2.13}$$

With this in mind, we can understand the motivation behind QAOA.

QAOA mimics an adiabatic evolution of $H(t) = (1-t)H_B + tH_C$. The Layers of QAOA correspond to trotterized segments of the time evolution operator. QAOA can be viewed as a trotterized quantum adiabatic algorithm. The optimization of the parameters γ, β can be viewed as finding a schedule for the adiabatic evolution. Since γ corresponds to t, we might expect a rising γ and a falling β. This is however not the case since we just try to find some good parameters, not specifically the ones that mimic the schedule.

2.3 MaxCut QAOA

This so far was the explanation for the general QAOA. What is still missing is, how QAOA can be used to solve the MaxCut problem. For this, one needs the cost operator which gives a cut value for some measurement. One operator that achieves this is the following.

$$C(z) = \frac{1}{2} \sum_{(i,j)\in E} (1 - Z_i Z_j) \tag{2.14}$$

Where Z_i is analogous to the mixing operator:

$$Z_j := I \otimes \cdots \otimes I \otimes \underbrace{Z}_{j} \otimes I \otimes \cdots \otimes I \tag{2.15}$$

The idea is, that a qubit is mapped to +1 or −1: $|0\rangle \mapsto +1, \quad |1\rangle \mapsto -1$. Therefore, if qubits q_i, q_j correspond to an edge and have different outcomes when measuring, the clause $\frac{1}{2}(1 - Z_i Z_j)$ evaluates to 1. If their outcomes are equal, it evaluates to 0. For the computation, this operator can be simplified as described in [12]:

(2.16)

$$\begin{aligned}
\langle C \rangle_{|\boldsymbol{\gamma},\boldsymbol{\beta}\rangle} &= \langle \boldsymbol{\gamma}, \boldsymbol{\beta}| \, C \, |\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle \\
&= \langle \boldsymbol{\gamma}, \boldsymbol{\beta}| \, \frac{1}{2} \sum_{(i,j)\in E} (1 - Z_i Z_j) \, |\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle \\
&= \frac{|E|}{2} - \frac{1}{2} \sum_{(i,j)\in E} \langle \boldsymbol{\gamma}, \boldsymbol{\beta}| \, Z_i Z_j \, |\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle \\
&= \frac{|E|}{2} - \frac{1}{2} \sum_{(i,j)\in E} e_{ij}(\boldsymbol{\gamma}, \boldsymbol{\beta})
\end{aligned}$$

Where e_{ij} is the contribution of an individual edge. Therefore, the cost operator used for the computation can be written as a sum of Pauli strings where each string corresponds to an edge in the graph.

$$C = \sum_{(i,j)\in E} Z_i Z_j \tag{2.17}$$

It should be noted, that an edge (i, j) induces a subgraph. To construct it, take the edge (i, j) and all edges that can be reached in p steps from (i, j). Add all the nodes such that each edge has endpoints. These subgraphs are of interest because it was shown in the original QAOA paper [11], that the expectation value can be evaluated as the sum over the subgraph contributions. These are the contributions e_{ij} from Equation (2.16).

2.4 WS-QAOA

The following is an explanation of WS-QAOA based on [10].

Equation (2.14) maps a node to the value -1 or $+1$ to calculate the objective function. Alternatively the objective function can be formulated as a so-called Quadratic Unconstrained Binary Optimization (QUBO).

(2.18)
$$C(x) = \frac{1}{2} \sum_{(i,j)\in E} \left(x_i \cdot (1 - x_j) + x_j \cdot (1 - x_i)\right)$$

In general, a QUBO takes the following form:

(2.19)
$$\min_{x\in\{0,1\}^n} x^T M x + m^T x \quad \text{or} \quad \max_{x\in\{0,1\}^n} x^T M x + m^T x$$

In this formulation, the variables x_i are binary $\{0, 1\}$. Since we can embed an NP-hard problem such as MaxCut into QUBO, solving QUBO problems is NP-Hard. One can relax the constraint of binary values $x_i \in \{0, 1\}$ to allow any value inside this interval $x_i \in [0, 1]$. The result of this continuous-valued relaxation is called Quadratic Programming (QP). Other transformations [21] yield a so-called Semidefinite Program (SDP). There exist solvers for both QP and SDP [13, 20]. Alternatively, one can use any algorithm that produces an approximate solution. This is the case for MaxCut, where the Goemans-Williamson (GW) algorithm can be used to generate a suitable approximate solution. For MaxCut this approximate solution is further called the *initial cut*. The idea of WS-QAOA is now, to compute such an approximate solution $c^* \in [0, 1]^n$ classically and use the result as an initialization for QAOA.

This is done by replacing the initial equal superposition state $|s\rangle$ with

(2.20)
$$|\phi^*\rangle = \bigotimes_{i=1}^{n} R_Y(\theta_i)\,|0\rangle$$

Where $\theta_i = 2 \arcsin\left(\sqrt{c_i^*}\right)$ and c_i^* is the i-th coordinate of the approximate solution. Notice that for Qubit i the initial state $|\phi^*\rangle_i$ is given by

(2.21)
$$|\phi^*\rangle_i = R_Y(\theta_i)\,|0\rangle = \sqrt{1 - c_i^*}\,|0\rangle + \sqrt{c_i^*}\,|1\rangle$$

Meaning, that the probability of measuring Qubit i as $|1\rangle$ is c_i^*.

The cost operator is left unchanged. The mixer however needs to be adapted. Recall, that the motivation of QAOA was to mimic AQC. This means that we need to evolve from an eigenstate of the mixer to the eigenstate of the cost operator. Therefore, the initial state that was just described, needs to be an eigenstate of the mixer. The mixer hamiltonian doesn't use the Pauli-X operator. Instead we have

(2.22)
$$M^{(i)} := \begin{pmatrix} 2c_i^* - 1 & -2\sqrt{c_i^*(1 - c_i^*)} \\ -2\sqrt{c_i^*(1 - c_i^*)} & 1 - 2c_i^* \end{pmatrix}$$

Analogous to X we use M to create M_j and the adapted mixer hamiltonian B'.

(2.23)

$$M_j := I \otimes \cdots \otimes I \otimes \underbrace{M^{(j)}}_{j} \otimes I \otimes \cdots \otimes I$$

(2.24)

$$B' := \sum_{j=1}^{n} M_j$$

It can be shown that $|\phi^*\rangle_i$ is an eigenstate of $M^{(i)}$.

(2.25) $M^{(i)} |\phi^*\rangle_i = - |\phi^*\rangle_i$

Expanding M_j and $|\phi^*\rangle$ as tensor products shows that

(2.26) $M_j |\phi^*\rangle = - |\phi^*\rangle$

With Equation (2.26) it is easy to see that $|\phi^*\rangle$ is in fact an eigenstate of B'.

(2.27) $\Rightarrow B' |\phi^*\rangle = \sum_{j=1}^{n} M_j |\phi^*\rangle = - \sum_{j=1}^{n} |\phi^*\rangle = -n |\phi^*\rangle$

With this, we get the adapted mixer

(2.28) $U(B', \beta) := e^{-i\beta B'} = \prod_{j=1}^{n} e^{-i\beta M_j}$

It can be implemented by using single-qubit rotations:

(2.29) $R_Y(\theta_i) R_Z(-2\beta) R_Y(-\theta_i)$

The circuit for WS-QAOA is shown in Figure 2.3.

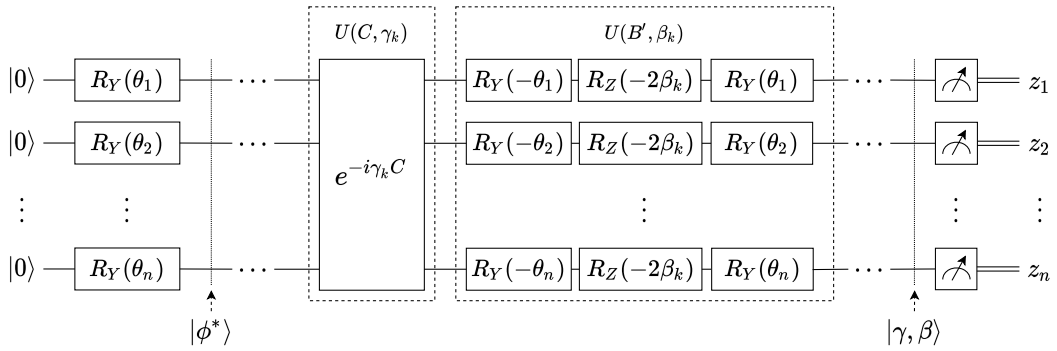

Figure 2.3: WS-QAOA circuit. Only one layer of cost operator and mixer is shown. The application of $U(C, \gamma)$ and $U(B', \beta)$ is repeated p times.

This approach has a small problem. If the cost operator only contains spin operators $Z_i Z_j$ like the one in Equation (2.17), a Qubit initialized to either $|0\rangle$ or $|1\rangle$ i.e. if $c_i^* = 0$ or $c_i^* = 1$, will remain in that state throughout the optimization. To mitigate this, a regularization parameter $\varepsilon \in [0, 0.5]$ is introduced. Instead of directly using c_i^* to calculate θ_i we replace c_i^* depending on its value: $\theta_i = 2 \arcsin \sqrt{x}$

$$x = \begin{cases} \varepsilon & \text{if } c_i^* \leq \varepsilon \\ c_i^* & \text{if } c_i^* \in [\varepsilon, 1-\varepsilon] \\ 1-\varepsilon & \text{if } c_i^* \geq 1-\varepsilon \end{cases} \tag{2.30}$$

Figure 2.4 illustrates the different cases.

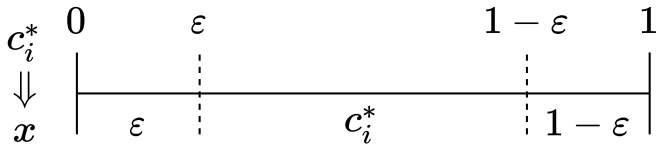

Figure 2.4: Choice of ε regularization. The bottom row gives the value of x, depending on where c_i^* falls in the interval.

The advantage of WS-QAOA is its performance at low depth. For small p (e.g. $p = 1$) WS-QAOA had an improved energy compared to standard QAOA. In further experiments (also at depth $p = 1$) it produced states, that were much closer to the minimum energy of the Hamiltonian compared to standard QAOA. Egger et al. [10] also looked at WS-QAOA in the context of Quantum Annealing (QA), which is a method to realize AQC (see AQC in Section 2.2) under less strict requirements [14]. They fixed an annealing schedule and compared the standard mixer Hamiltonian to the warm-start mixer. The warm-start mixer required less annealing time and the final energy was lower than for the standard mixer. They concluded, from an annealing point of view, that WS-QAOA converges faster than QAOA.

3 Motivation

Quantum computation is a field of study that has seen an increasing effort in research. It promises to outdo classical computing devices at least for some problems. One of the major breakthroughs was in 1994 when Peter Shor proposed an efficient algorithm that could theoretically factor large numbers [23]. There is no known algorithm to date that solves this problem efficiently on a classical computer.

However, Shor's algorithm requires an ideal quantum computer. Meanwhile, the biggest problem that quantum computers face today is noise. This noise has several causes: Qubits are unstable and the state of a qubit decays over time, which is known as decoherence. Gates aren't 100% accurate, every gate introduces errors. In a circuit with high depth, the error accumulates with each gate, meaning, the higher the depth the greater the error [9]. Readout errors appear when measuring the result of a quantum computation since the measurement takes time in which the state can decay. Therefore, too much noise causes the computation to become faulty and results to become incorrect. This is why current quantum computers are called Noisy Intermediate Scale Quantum (NISQ) devices. There are several approaches ranging from mitigating errors by post-processing the measurement, trying to prevent errors by adapting circuits to fit the hardware or reducing errors during the computation by employing error correction techniques [2]. The last approach needs a lot of physical qubits and for this reason, it is currently infeasible. Alternatively one can work on the hardware or software, meaning one can either work to improve the fidelity of the operations or develop algorithms that are robust enough to tolerate such deviations. For the near term, QAOA is a very promising candidate for an algorithm that might cope with these difficulties. Its depth is parameterized by a number p and can be adjusted, sacrificing the optimality of the result for a shallower depth. This relatively shallow depth of QAOA, compared to other algorithms, like the HHL algorithm [17] for solving linear systems of equations or Shor's algorithm, helps to avoid the accumulation of errors, introduced by gates or an increase of computation time.

The general procedure of QAOA consists of two parts. The execution of a parameterized circuit and optimization of said parameters. In the most basic case, random initial parameters are chosen and the result of the execution is measured. An optimizer then tries to adjust the parameters to get a better result. This is done until some convergence criterion is reached. In an attempt to get the global optimum, several optimization runs are performed with different random initializations. This basic approach takes a lot of calls to the quantum hardware. Ongoing research into QAOA has yielded ideas to improve this execution. These ideas are described in Chapter 4. Since a key aspect of QAOA is to find good angles $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$, these works investigate how this process can be optimized, for example by finding a good initialization for these angles [1] or transferring the values between problem instances [12]. Possessing good initial parameters can help to reduce the number of calls to the quantum hardware and aid developers in finding optimal ones. However, these investigations concern the standard version of QAOA [11]. [12] justified a transfer of parameters by analyzing the contribution of subgraphs to the overall energy. As opposed to QAOA, WS-QAOA has different subgraph structures, mainly warm-starting completely changes the contributing subgraphs. For this

reason, it is unclear if the approach works for WS-QAOA. As explained in Section 2.4, WS-QAOA comes with improvements to QAOA, mainly its performance with fewer layers. Combining these advantages with the ability to find good parameters might increase the potential of WS-QAOA.

I investigate if and how known methods for determining good initial parameters can be applied to WS-QAOA. In particular, I analyze the subgraph structures of warm-started regular graphs to derive patterns on when a transfer can be successful. I explore if the space of parameters exhibits a concentration of good parameters, i.e. are the optimal parameters of different WS-QAOA instances located close to each other. A different view on this would be to see if good parameters for one instance also constitute good parameters for another related instance. This ties into the transferability of parameters. There has been recent work [12] that established conditions indicating when a transfer of QAOA parameters yields good results. Whether or not these conditions hold for the warm-started variant is a topic of this thesis. The goal is, that these insights can be used to reduce the number of evaluations in the optimization loop or to calculate good parameters without running the optimization loop at all.

4 Related Work

There has already been some research into the parameters of QAOA. This includes ideas for finding good initializations for the parameters [4, 8, 18], transferring them directly [12, 22] or their concentration [1, 5]. The most relevant ideas upon which this thesis builds are described in the following sections.

4.1 Parameter Transferability Between Random Graphs

Galda et al. [12] investigated under what conditions the optimized QAOA parameters for one graph also maximize the QAOA objective function for another graph. To see when and why a transfer of optimal QAOA parameters is possible, they analyzed the transferability of subgraphs. This was done by generating so-called *landscapes* of energy contributions. They noticed, that the locations

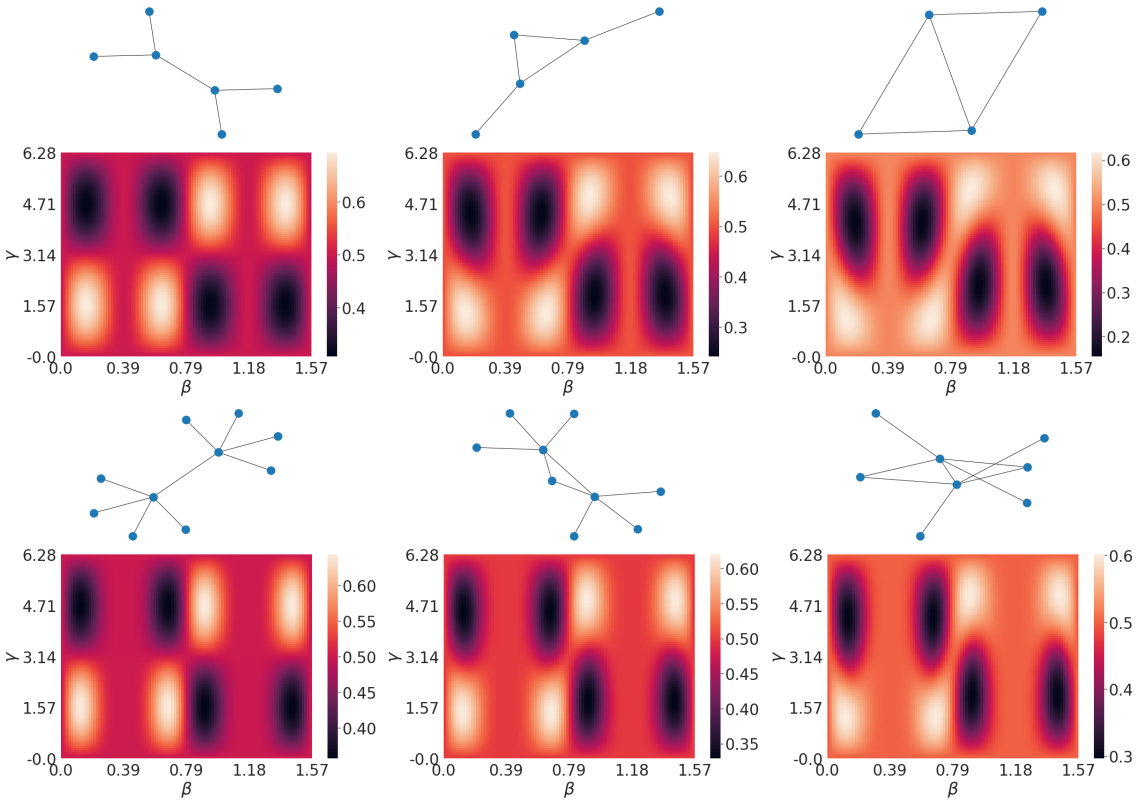

Figure 4.1: Selection of subgraphs (3- and 5-regular) and their corresponding landscapes by Galda et al. [12].

of the maxima of the subgraphs of a d-regular graph, $d \leq 8$ coincided with each other. See for example the subgraphs of a 3-regular graph in the first row of Figure 4.1. Additionally, the location of maxima of odd-regular graphs coincide with each other as well as those of even-regular graphs. This is demonstrated in Figure 4.1 with subgraphs of 3-regular graphs in the top row and a selection of subgraphs of 5-regular graphs at the bottom.

This means, for example, that optimal parameters can be transferred between any two instances of 3-regular graphs and remain relatively optimal. From this, they formulated three sufficient conditions for optimal parameter transferability: Optimal QAOA parameters are

- mutually transferable between all subgraphs of the donor graph;
- mutually transferable between all subgraphs of the acceptor graph;
- transferable from every subgraph of the donor graph to every subgraph of the acceptor graph.

Further, they created a map that visualizes the parameter transferability between subgraphs of regular graphs, where regular graphs up to degree $d = 8$ were considered. From regular graphs, they further investigated random graphs. To be more precise, they looked at every possible subgraph with a maximum node degree of $d = 6$ and created a transferability map for subgraphs of random graphs.

They used these maps to look for small donor graphs with 6 nodes and transferred the optimized parameters of them to graphs with 64 nodes. The transfer was successful in the sense, that the approximation ratio for the transferred parameters differed from the approximation ratio of the optimal parameters by just about 2 percentage points or less.

4.2 Concentration for Fixed Control Parameters

Another paper [5] that analyzed the distribution of parameters also used MaxCut on 3-regular graphs to evaluate if the objective function value concentrated if the parameters are fixed. This means that typical instances have (nearly) the same value of the objective function for the same parameters. Among other analyses, Brandao et al. [5] supported their findings with numerical examples.

To be precise, they first optimized the parameters for a small random instance (a 3-regular graph with 10 nodes) and used the parameters that were found for 25 instances with 20 nodes. They analyzed the performance of the optimized parameters at depths $p = 2 \ldots 7$. At $p = 2$ the mean approximation ratio was $\frac{22.409}{26} = 0.8619$, at depth $p = 7$ it increased to $\frac{25.110}{26} = 0.9658$.

For the depth $p = 8$ the optimized angles resulted in an approximation ratio of 0.984 for the 10 bit instance. After transferring the parameters to a random instance with 24 nodes, the mean approximation ratio calculated from 25 random instances was 0.934 with a standard deviation of 0.014.

5 Experiment Setup

A lot can be learned by looking at the values of the objective function. The first section of this chapter describes how this objective function is visualized. As this visualization is a vital part of the analysis it is worth describing the details of its creation. The second section describes how properties of QAOA can be leveraged to simplify calculations when dealing with QAOA.

5.1 Generating Landscapes

Energy landscapes are a nice way to visualize the effect of different parameters on the expectation value. When we set $p = 1$, we only have two parameters γ, β. With these, we can create a 2D plot, where the x- and y-axis represent the angles β, γ respectively and the color of a specific point indicates high or low energy. Some implementations use the negative value of the cut as the objective function [27]. This is done to state MaxCut as a minimization problem and use common optimization tools. To be consistent with Galda et al. [12], MaxCut is stated as a maximization problem. Since the goal is to maximize the energy, a bright spot indicates high energy and therefore good parameter values, and a dark spot indicates low energy and thus bad parameter values. An example of such a landscape is shown in Figure 5.1.

One may notice, that the energy landscapes are transposed, i.e. the axes are flipped, compared to Figure 4.1 [12]. This is likely due to a difference in the implementation. Additionally, β runs from 0 to π instead of $\beta \in [0, \frac{\pi}{2}]$. This makes a difference when we consider WS-QAOA for these graphs.

With these plots, it is easy to compare different graphs in terms of their parameters. If there are lots of places, where the maxima coincide, then the parameters that give a high expectation value for one graph should also give a high expectation value for the other.

The angles γ, β are bound by different ranges. We have $\gamma \in [0, 2\pi]$ and $\beta \in [0, \pi]$ as we are dealing with unweighted edges. The parameter space, in general, is $[0, 2\pi]^p \times [0, \pi]^p$. For example, 100 samples for each parameter (γ, β) were used to generate the energy landscape shown in Figure 5.1.

The procedure of generating the data for such an energy landscape is best described with an example. First, choose the graph that should be analyzed. This example uses the graph shown in Figure 5.1.

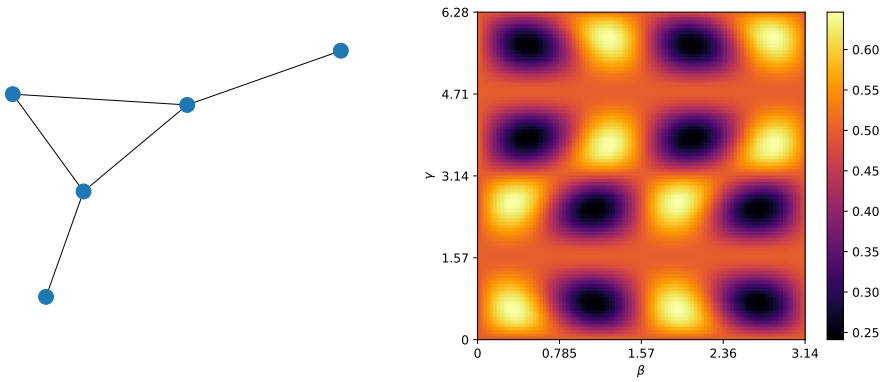

Figure 5.1: Energy landscape for one of the subgraphs of 3-regular graphs. The energy refers to the central edge. High energy means that it is likely, that the central edge is part of the cut i.e. the corresponding nodes are in different partitions.

From this create the operator as described in Section 2.3:

$$\begin{aligned} C = -\frac{1}{2}\,(&I \otimes I \otimes I \otimes Z \otimes Z \\ &+I \otimes I \otimes Z \otimes Z \otimes I \\ &+I \otimes Z \otimes I \otimes Z \otimes I \\ &+I \otimes Z \otimes Z \otimes I \otimes I \\ &+Z \otimes Z \otimes I \otimes I \otimes I) \end{aligned} \tag{5.1}$$

This operator can be used to create a QAOA circuit. A software development kit that can create such circuits is Qiskit [26]. It also has several algorithms already implemented. The class `qiskit.algorithms.QAOA` has a method `construct_circuit` that takes an operator like C and parameters γ, β to construct the QAOA circuit. Transpiling this circuit with the gates `['h', 'cx', 'rz', 'rx']` gives the circuit depicted in Figure 5.2. These are commonly used gates and make for a relatively simple circuit. This circuit can then be executed on a QPU or it can be simulated on classical hardware with the `'statevector_simulator'`. This simulator has the advantage of providing exact results without any noise or multiple shots necessary to generate a stochastic

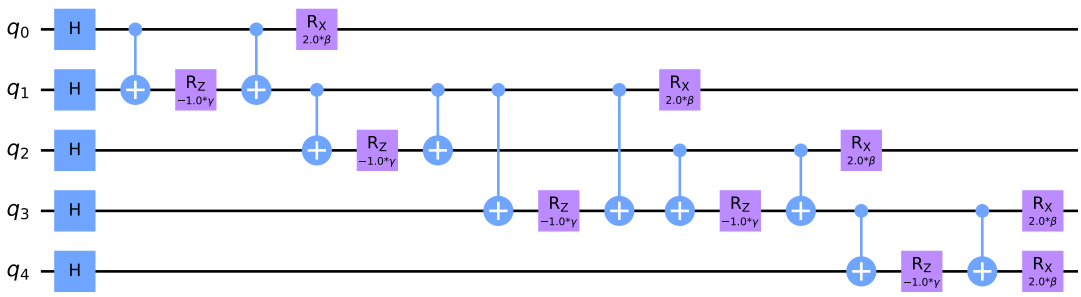

Figure 5.2: QAOA circuit for the graph of Figure 5.1.

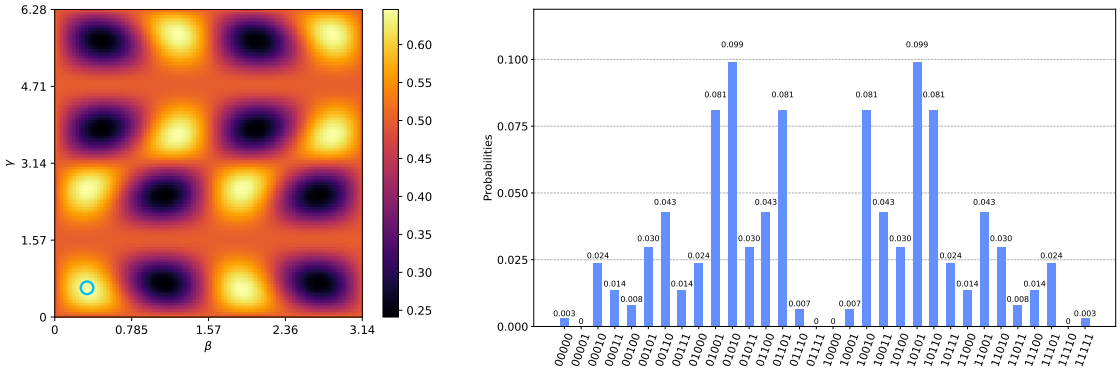

Figure 5.3: Energy landscape with maximizing parameters $\gamma \approx \frac{9\pi}{50}$, $\beta \approx \frac{\pi}{10}$. Resulting probability distribution when running the circuit of Figure 5.2 with these parameters in the `statevector_simulator`.

distribution. Therefore, one obtains a distribution such as Figure 5.3 that gives the exact probability of measuring a particular state. Using the value of a cut, one can easily calculate the expected value for given parameters. Doing this for a lot of samples creates the energy landscape. One important detail is, that if a subgraph is analyzed, the expectation value is only calculated for the "central" edge. The central edge in this example is running between nodes 1 and 3. One can read from Figure 5.3 that `01010` and `10101` have the highest probability of being measured. These don't cut the central edge. However, other likely results, such as `01001`, `01101`, `10010` and `10110` do. Qiskit [26] uses little-endian ordering for qubit ordering, meaning a cut like `01001` refers to the values of nodes `4,3,2,1,0`, i.e. nodes `4,2,1` are in one partition and `3,0` are in the other one. Sometimes a cut is written in brackets: `[1 0 0 1 0]`, this cut has the reverse order and refers to nodes `[0 1 2 3 4]`.

Doing this with all possible subgraphs of 3-regular graphs (see Figure 5.4) gives comparable results to Section 4.1.

To generate the landscape for WS-QAOA one has to adapt the circuit to employ the technique of warm-starting. Namely, the initial state and the mixer are changed as described in Section 2.4. As an example see Figure 5.5.

5.2 Subgraph Optimization

For calculating the energy only the central edge is taken into account because the total energy of QAOA can be calculated over just the contributions of the subgraphs as mentioned in Section 2.3. This fact makes finding optimal parameters for depth $p = 1$ trivial in some cases. For example, finding the optimal parameters for any 3-regular graph can be achieved as follows:

1. Calculate the three landscapes belonging to the three types of subgraphs.
2. For each type, count the number of occurrences of each subgraph.
3. Calculate the total energy by adding the landscapes, weighted by their occurrence.
4. Read the optimal parameters from the maximum of this landscape.

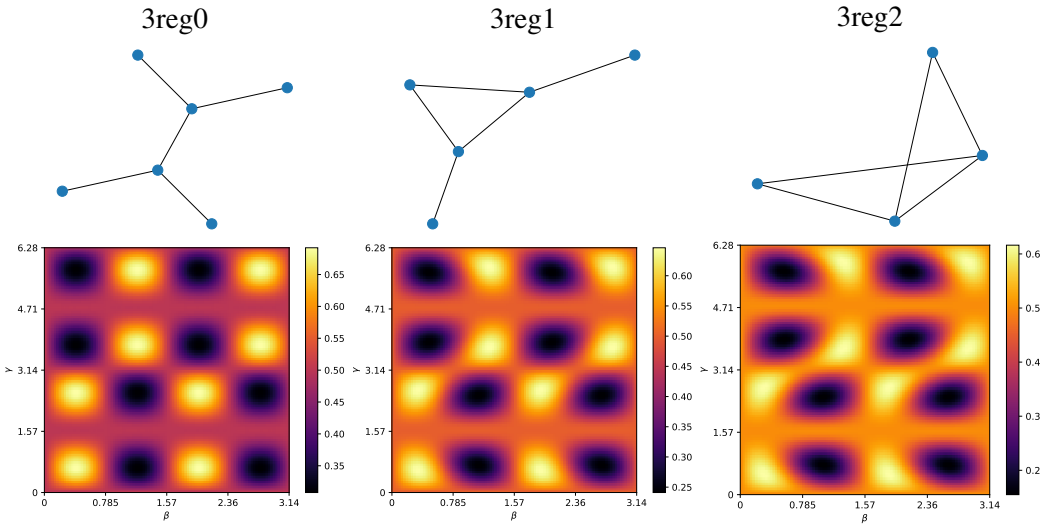

Figure 5.4: All possible subgraphs of a 3-regular graph. The central edge is the one where its two endpoints have degree 3. The subgraphs are named for later reference. Below each graph is its corresponding energy landscape.

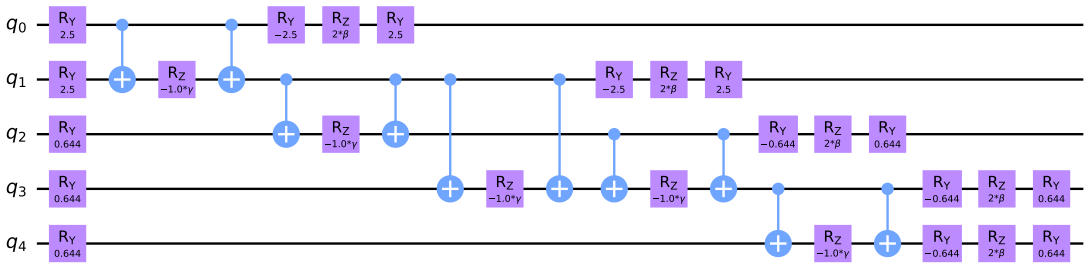

Figure 5.5: WS-QAOA circuit for the graph of Figure 5.6.

If the situation is more complex, e.g. not having a 3-regular graph but a random graph, instead of calculating the whole landscape, the optimization can be performed on the subgraphs. To be more precise, the energy for each subgraph is calculated and the contribution of every central edge is added together. This reduces the number of qubits necessary in the optimization loop. Such an optimization can be seen in Figure 5.7. Not all optimizations find a global optimum. This technique of only considering the subgraphs also saves computation time as results such as the energy calculated from a subgraph can be reused. In Section 6.4 this technique is used to determine optimal parameters. To determine a cut, however, the circuit for the whole graph has to be run.

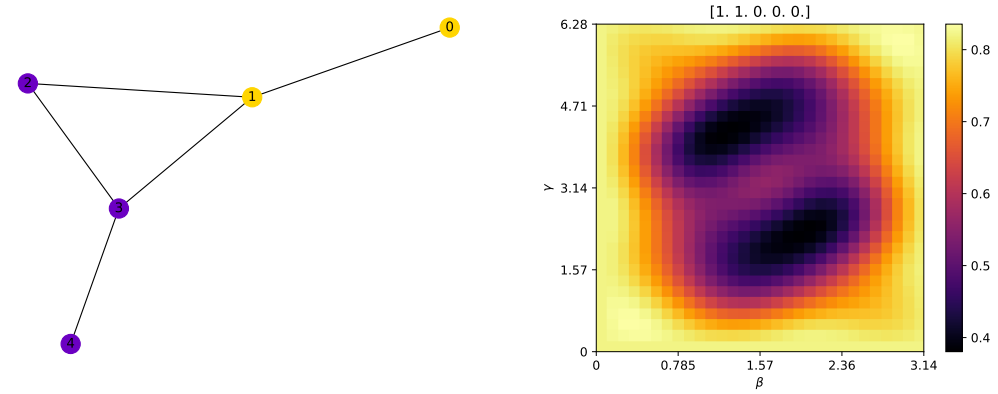

Figure 5.6: Energy landscape for a warm-started subgraph of a 3-regular graph. The title of the landscape is the initial cut `[1. 1. 0. 0. 0.]`, where the leftmost value corresponds to node 0. The energy refers to the central edge. The color of a node indicates its initialization: purple 0, yellow 1.

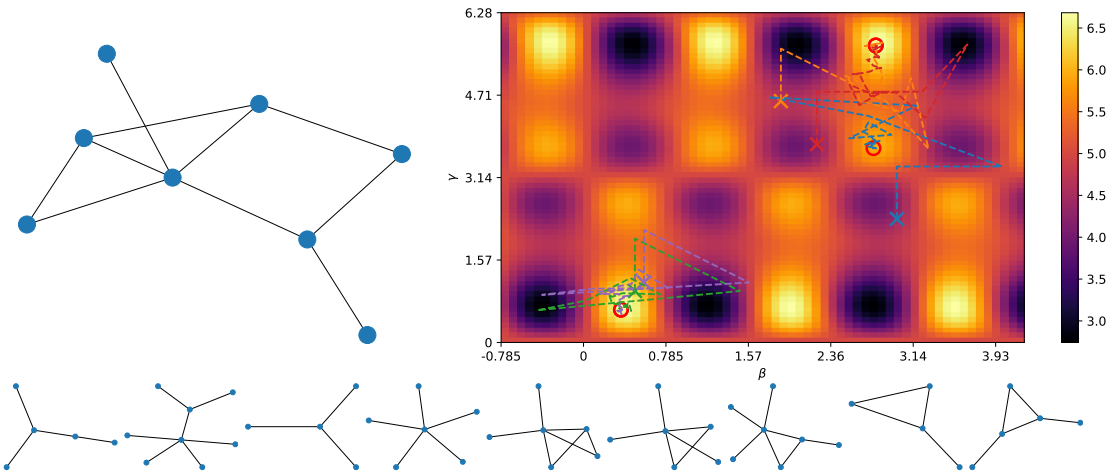

Figure 5.7: Several optimizations performed on the subgraphs using the COBYLA optimizer. The ×-symbols denote the randomly chosen initial parameters, the red circles are the optima that are found. Displayed is the random graph with its subgraphs (bottom row), the first subgraph appears twice.

6 Parameter Transferability for WS-QAOA

One way to initialize the parameters is to take good parameters of other instances and reuse them. Generally, one can not expect good results when parameters are transferred between arbitrary instances. But there might be some conditions under which a transfer may be successful. This chapter analyzes patterns between related instances and when parameters can be successfully transferred if at all. The first section describes the influence of the hyperparameter ε. Followed by Section 6.2 which is the main analysis of regular graphs up to degree $d = 5$. The results and insights are summarized in Section 6.3 in a visualization called transferability map. The next section investigates if the results transfer to larger instances by analyzing parameter concentration of warm-started regular graph intances. Lastly, Section 6.5 describes how the insights can be used to precompute parameters when warm-starting is employed.

6.1 Influence of ε

To get a better grasp on warm-starting and its effects, let's take a closer look at the influence of the hyperparameter ε. This regularization parameter can be viewed as a continuous mapping between WS-QAOA and standard QAOA. For $\varepsilon = 0.5$ the angle $\theta_i = 2 \arcsin\left(\frac{1}{\sqrt{2}}\right) = \frac{\pi}{2}$ transforms the initialization of one qubit into the equal superposition: $R_Y\left(\frac{\pi}{2}\right)|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. In other words, the same initial state as standard QAOA. And, apart from a sign change, the mixer is also the same. So, the lower the ε, the more warm-starting we use. This effect is also reflected in the energy landscapes in Figure 6.1. The last landscape ($\varepsilon = 0$) shows what fully warm-starting would look like. Here, the energy is fully determined by the classically precomputed solution, $\boldsymbol{\gamma}, \boldsymbol{\beta}$ have no influence. Since the central edge is part of the cut, the edge contributes an energy of 1.

In the later analysis, a value of $\varepsilon = 0.1$ is chosen. Also, p is fixed at $p = 1$ which is required for the analysis with the 2D landscapes. This however isn't too big of a restriction, as WS-QAOA shows good performance at already low depth. Analysis with different ε and $p > 1$ is left for future works, however, the methods are the same and similar behavior is conjectured.

6.2 Warm-starting of Regular Graphs

3-regular Graphs

Now let's take a closer look at the actual effect of warm-starting. The goal is to see whether or not the claims of Galda et al. [12] also hold for warm-started graphs. For this, I took a look at all the possible initial cuts for 3-regular subgraphs. The possible subgraphs of 3-regular graphs are shown in Figure 5.4

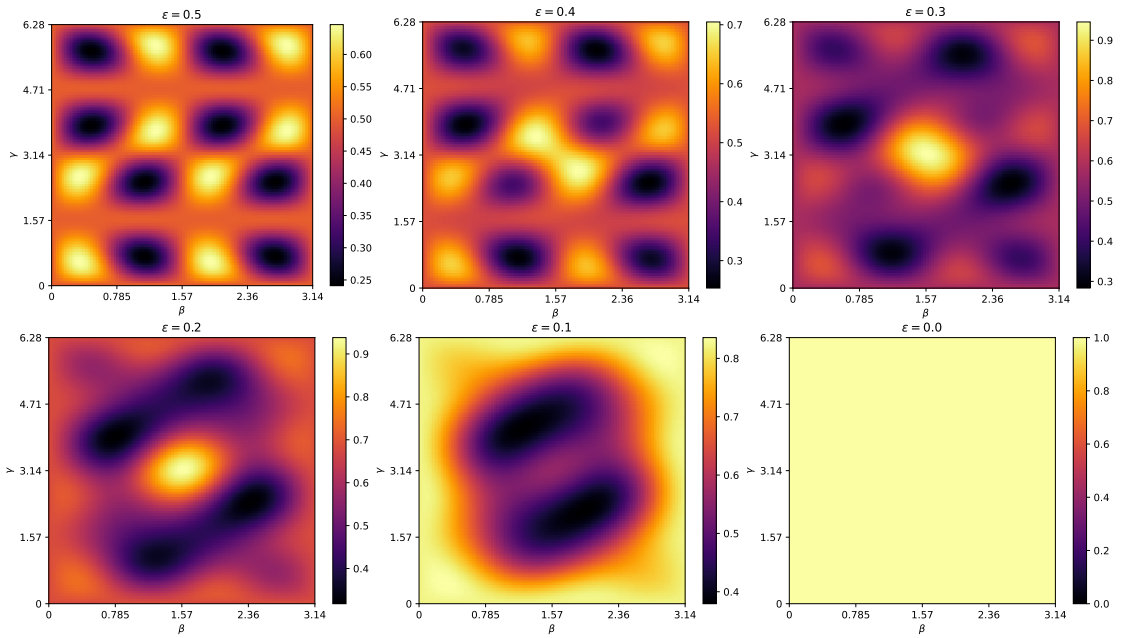

Figure 6.1: Landscapes for the initial cut `[1 1 0 0 0]` (see Figure 5.6) with different ε. The top left ($\varepsilon = 0.5$) is the same landscape as if no warm-starting is employed. Because the scale in each landscape is relative, some regions only appear to get darker while the energy increases, notice the colorbars.

At first, a node could be initialized with either 0, 0.5 or 1. For a simple subgraph with n nodes, this means 3^n possible initial cuts. The graph 3reg1 for example would have $3^5 = 243$ initial cuts. Since this number is growing so fast, making analysis more difficult, only initial cuts containing 0 and 1 are considered. This drastically reduces the number of initial cuts to $2^5 = 32$. A big part of generating all possible initial cuts is figuring out which initial cuts are equivalent to each other. For example the resulting energy of the initial cut `[0 1 0 0 1]` is the same as its inversion `[1 0 1 1 0]` since the value of the cut is the same regardless of the underlying graph. For our purposes, equal energy means an equal energy landscape.

But the graph structure can also be taken into account. For the 3reg1 graph the initial cuts `[0 1 0 0 1]` and `[1 0 0 1 0]` are equivalent. The initial cut can be viewed as a coloring of the graph and the graphs resulting from this coloring are isomorphic. This means both energy calculations are of the same graph, thus their energy is equivalent.

After eliminating all those "duplicated" landscapes we are left with 12, 10 and 5 initial cuts for 3reg0, 3reg1 and 3reg2 respectively. All initial cuts generate a landscape using the procedure described in Section 5.1. These landscapes are shown in Figures 6.2, 6.4 and 6.5.

The first thing to notice in Figure 6.2 is, that the landscapes look significantly different from the ones where no warm-starting was employed (cf. Figure 5.4). A maximum for one initial cut might be a minimum for another. This is the case for the outermost landscapes at the bottom of Figure 6.2. Parameters located at the edge of the plot correspond to a minimum (leftmost) or a maximum

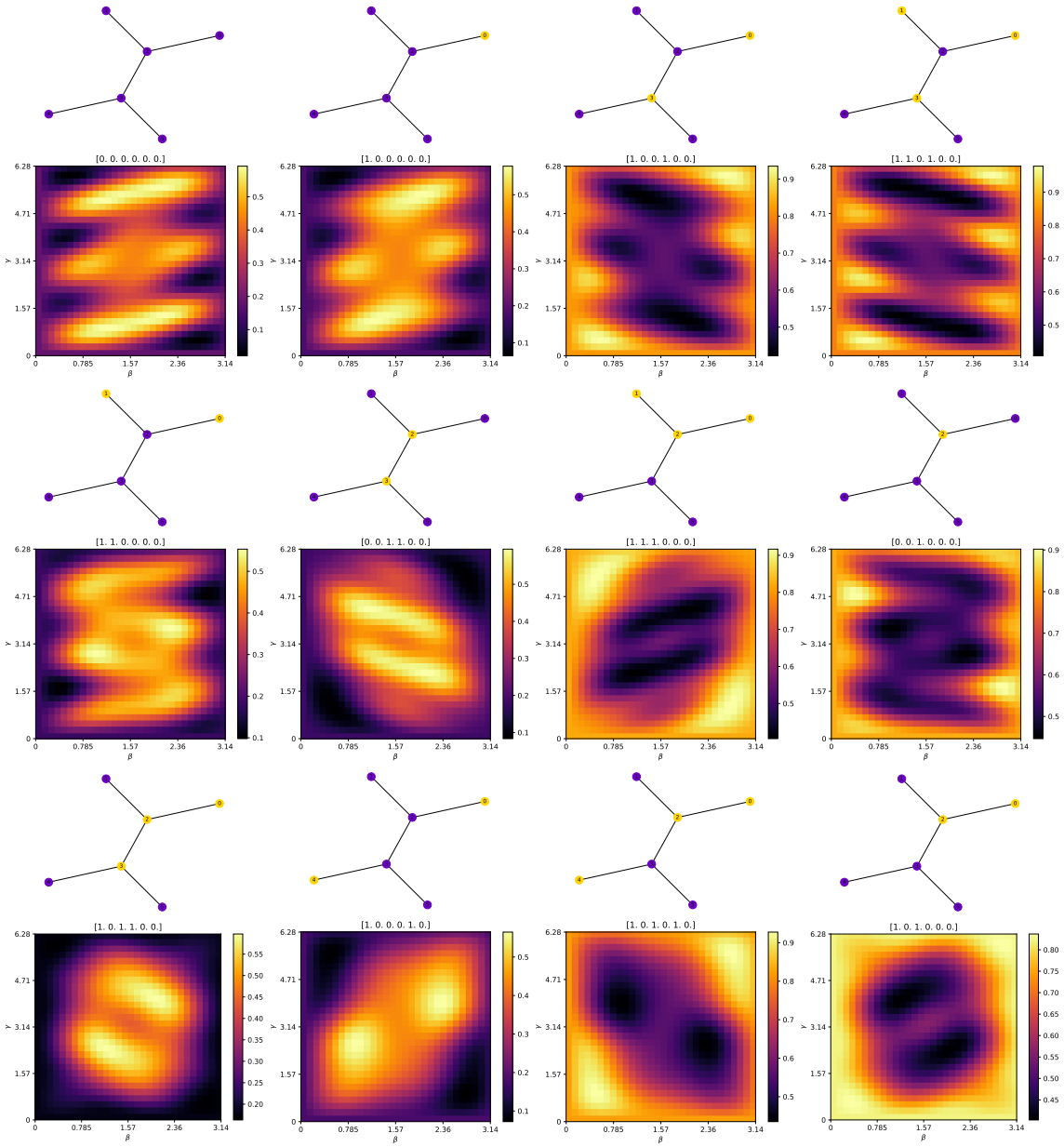

Figure 6.2: All 12 initial cuts for 3reg0. On top of each landscape the, warm-started subgraph is depicted. The images are arranged, such that the mirrored relationship is visible.

(rightmost). This means that transferring parameters between warm-started 3-regular graphs is not easily justifiable, since even the *same* graph with different initial cuts has vastly different landscapes and maxima don't necessarily coincide.

Another thing that becomes apparent, is the landscapes on the left have low energy around the edges of the plot, while the center shows higher energy. For the landscapes on the right, it is the other way around, with high energy on the sides and low energy at the center. All the graphs on the left have an initial cut where the central edge isn't part of the cut i.e. the central two nodes are initialized with the same value. Since the energy is mainly influenced by this edge, the bad initial cut creates

low energy. The graphs on the right have an initial cut where the central edge is part of the cut, the central two nodes are initialized differently. This seems to be the reason for the high energy around the sides. Intuitively this makes sense, take $\gamma = 0, \beta = 0$. In this case, the cost operator and mixer have no effect and the initialization determines the energy. A bad initial cut should give low energy, a good one should result in high energy. This is also reflected in the absolute energy. The colorbar to the right of each plot has a way higher maximum for the landscapes on the right. The maximum energy of a bad initial cut is close to the minimum energy of a good warm-start.

Another pattern that appears is a mirrored relationship between the landscapes. The landscapes appear to be inverted and flipped around the x-axis or the y-axis. The pattern that connects two landscapes is the following: Take any node of the central edge and invert all its neighbors by exchanging a 0 for a 1 and vice versa. Keep in mind that equivalent initial cuts result in equal landscapes.

As a reason for this behavior one can take a closer look at the probability distributions of the two landscapes. The parameters for the mirrored landscape are reflected across $\beta = \frac{\pi}{2}$. So, we have to look at the distribution for the two pairs (γ, β) and $(\gamma, -\beta + \pi)$. An example of these two distributions is depicted in Figure 6.3. The distributions are identical in their probabilities but differ in their measurements. In this example, one can get the measurements for the other if the measurements are XORed with `001011`. In general: for two initial cuts c_0, c_1 the measurements m_0, m_1 with $m_0 = m_1 \oplus c_0 \oplus c_1$ have the same probability. A concise mathematical proof of this fact is left for future works. Since the energy is determined by the central edge, the two distributions give opposing energies resulting in mirrored and inverted landscapes.

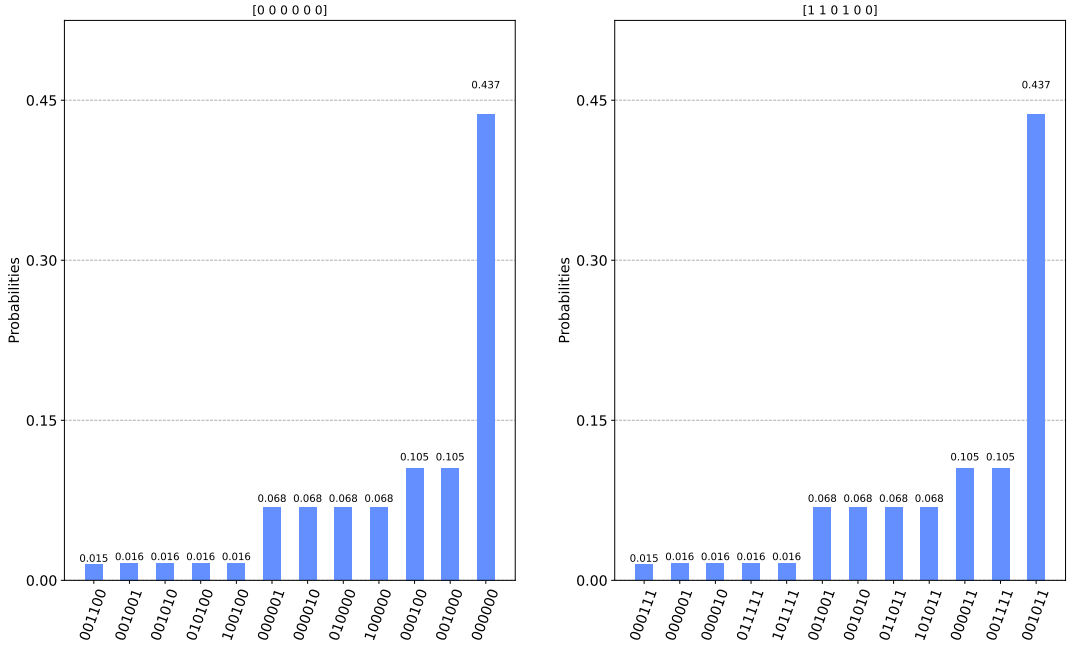

Figure 6.3: Distribution for the initial cuts `[0 0 0 0 0 0]` and `[1 1 0 1 0 0]` (see Figure 6.2 top left and right). Left distribution used $\gamma = 0.4$, $\beta = 0.2$, right had the parameters reflected at $\beta = \frac{\pi}{2}$ giving $\gamma = 0.4$, $\beta = -0.2 + \pi$. Shown are only measurements where the probability is above 0.01. Results are identical if each measurement is XORed with `001011`.

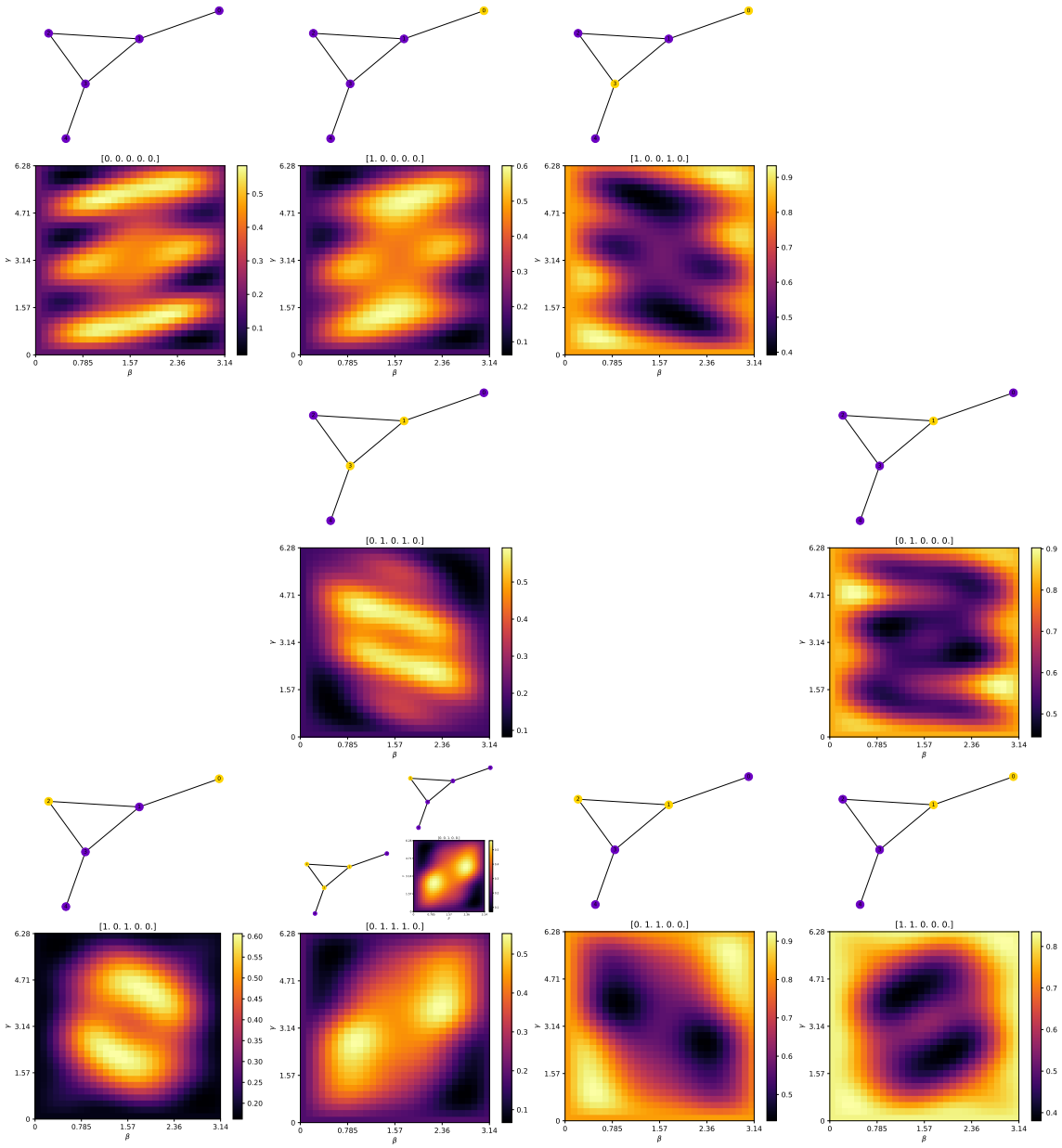

Figure 6.4: All 10 initial cuts for 3reg1. Empty spots where no mirrored landscape could be found. The middle row has an interesting case, where two landscapes look identical while having different initial cuts.

It is also worth mentioning, that changing the value of one outer node doesn't affect the landscape as drastically as changing one of the central nodes. For example, the first two landscapes in the top row of Figure 6.2 look quite similar.

Now let's take a closer look at the landscapes for 3reg1 (Figure 6.4). The landscapes are arranged in the same way as in Figure 6.2 to highlight the similarities of the landscapes between 3reg0 and 3reg1. This is also an important observation: corresponding initial cuts produce similar landscapes for 3reg0 and 3reg1. What exactly does it mean for two initial cuts to correspond to each other?

3reg1 has one node less than 3reg0. An initial cut of 3reg0 can't be applied to 3reg1. However, for each initial cut of 3reg1, there is a corresponding initial cut for 3reg0, which is in some sense equal. An easy example is the initial cut `[0 0 0 0 0]`. The corresponding initial cut of 3reg0 is the one containing all zeros `[0 0 0 0 0 0]`. The similarity of those landscapes is clearly visible. In the more general case, to find the corresponding initial cut it helps to think of the 3reg1 graph as an instance of 3reg0 where two of the outer nodes have been fused together. So to get back from 3reg1 to 3reg0, one has to split the node that is connected to the two central nodes. In this case the "tip of the triangle", node 2. The "new" nodes get the same initialization as the one that was split in two. This concept is meant when two warm-starts of different types of subgraphs are considered "equal". Apart from minor distortions, these "equal" landscapes look nearly identical.

This correspondence also explains, why some landscapes are "missing" in Figure 6.4. Not all initial cuts for 3reg0 can be obtained by splitting the node of 3reg1. Take the initial cut for the missing entry of the top row. The graph would need to have outer nodes that have opposite initializations and after splitting the nodes also need to have opposing initializations This is not possible. The initial cut `[0 0 0 0 0]` therefore doesn't have a mirrored landscape. To find the mirrored landscape of a 3reg1 graph, one has to determine the corresponding 3reg0 graph and invert the neighbors of a central node. Finally, merge two of the outer nodes that have the same warm-start.

An interesting thing that happens in the case of 3reg1 is also shown in 6.4. There are initial cuts where the graphs aren't equivalent to each other yet their landscapes are virtually identical (bottom row, center left). But with the concept of splitting nodes, this phenomenon is easily explained. If we were to split the relevant node, the resulting graphs would be equivalent to each other again.

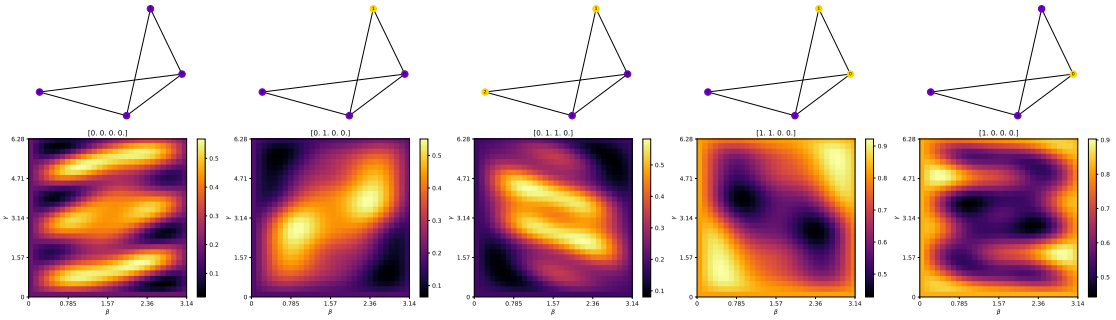

Figure 6.5: All 5 initial cuts for 3reg2. Only one landscape has a mirrored counterpart.

For the 3reg2 graphs, the same things can be said as for the 3reg1 graphs. The concept of splitting and merging nodes to get from one subgraph type to the other can also be applied. This explains why there are even fewer initial cuts and even more are missing their mirrored counterpart. While the distortions are a bit more visible than for the 3reg1 case, the similarity of "equal" initial cuts is clearly visible.

In conclusion, with the idea of equality between the initial cuts, the same effects appear analog to the ones described in Section 4.1: Subgraphs of 3-regular graphs have maxima in the same regions if their initial cuts are "equal".

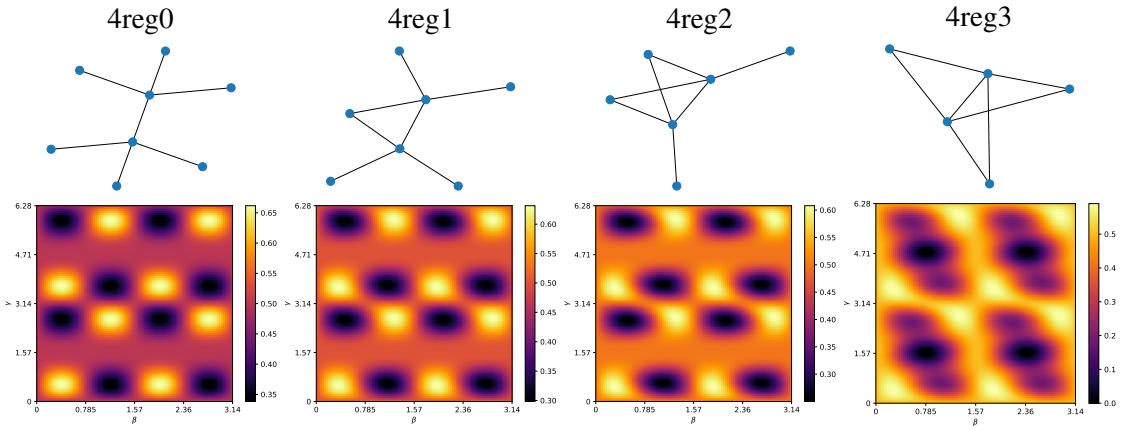

Figure 6.6: All possible subgraphs of a 4-regular graph. The central edge is the one where its two endpoints have degree 4. The subgraphs are named for later reference. Below each graph is its corresponding energy landscape.

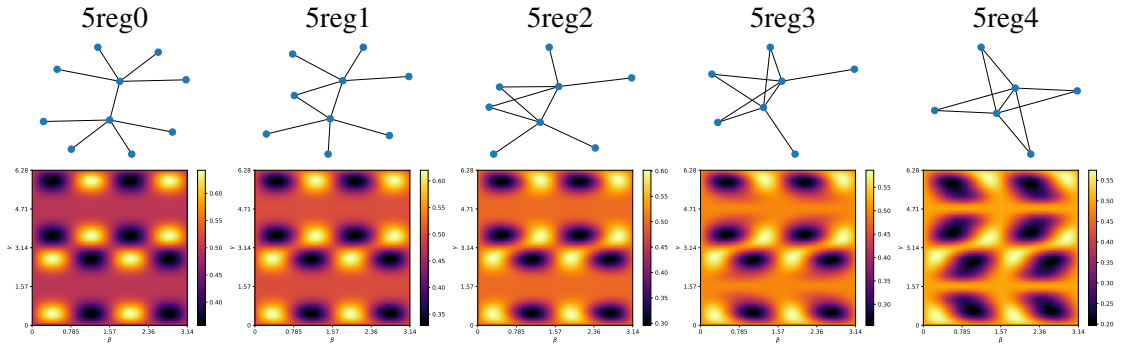

Figure 6.7: All possible subgraphs of a 5-regular graph. The central edge is the one where its two endpoints have degree 5. The subgraphs are named for later reference. Below each graph is its corresponding energy landscape.

4- and 5-regular Graphs

In the next step higher regular graphs are analyzed. Basically, all of the effects described for 3-regular graphs also apply to 4- and 5- regular graphs. Mainly the mirrored landscapes and similarities within a d-regular graph ($d \in \{4, 5\}$) appear as well. A complete list of all landscapes for 4- and 5-regular graphs can be found in Appendix A (Figures A.1 to A.9).

The landscapes for the 4-regular type differ from the 3-regular ones. It looks like the landscapes repeat already after $\gamma = \pi$ instead of 2π. Another thing that is apparent with that many initial cuts is that the type where no nodes are merged (3reg0, 4reg0, 5reg0) is the most general in terms of different landscape structures. The other types don't add any new landscapes.

The new landscapes don't offer new insights into the warm-starting of regular graphs in general. More interesting is the relationship among them. In the cold-started case, comparing Figure 5.4 and Figure 6.7, the location of the landscapes' maxima appear close to each other. These findings

suggest that there might be similar landscapes between 3- and 5-regular landscapes. For example, one might expect that the landscapes for similar warm-starts also look similar. To see if this holds, let's look at some examples.

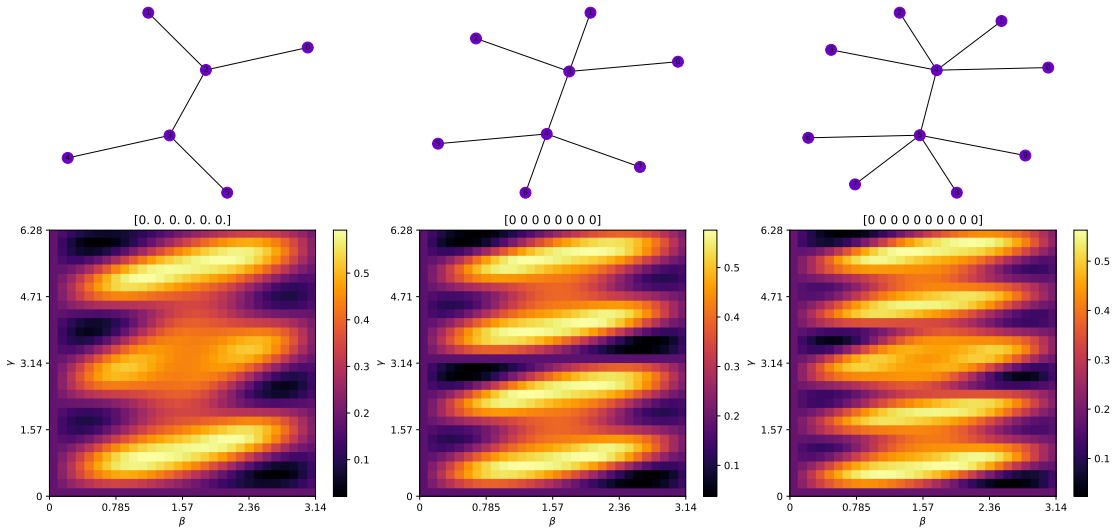

Figure 6.8: Subgraphs where every node is initialized with 0.

Figure 6.8 shows the landscapes where every node is initialized with 0. There are some similarities in the shape of the landscapes. A feature that appears in all 3 figures is a maximum whose shape resembles an ellipse. These ellipses are more squashed together for the 4 and 5 regular graphs. While the center of the 4-regular landscape has low energy, the 3 and 5 regular ones have high energy at this location. Interestingly, the 3-regular landscape seems to reappear in the 5-regular landscape. It seems as if the 3-regular landscape was scaled down along the y-axis by a factor of 2 and centered at $\gamma = \pi$. This pattern can also be observed for other landscapes (compare Figure 6.2 and Figure A.5).

To get back to the question of the landscape's similarity and location of their maxima, it does not appear as obvious as in the cold-started case. While the 3- and 4- regular landscapes differ in some locations, especially the center, the 3- and 5-regular ones also seem to have maxima at different locations. It might be possible to take the parameters of the 3-regular landscape and transfer them to the 5-regular while remaining close to a maximum. The other way around, however, doesn't look promising. For example, taking a maximum from the second ellipse from the top of the 5-regular landscape and using these parameters on the 3-regular landscape doesn't land in a maximum. A more precise analysis of this fact is performed in Section 6.3.

While this intuitive approach doesn't seem too promising, something rather surprising happens when we look at other 5-regular landscapes (Figure A.5). Namely, some of the landscapes look virtually identical to the 3-regular ones. The two bottom rows contain all the different landscapes that appear in the 3-regular case. All these initial cuts have something in common. Each central node has a neighbor that is initialized to 0 and one neighbor that is initialized to 1. If we were to remove these 4 nodes from the graph, the resulting subgraph would be the same as the 3-regular version that has this landscape. For these cases, good transferability is expected. An example of such landscapes can be seen in Figure 6.9.

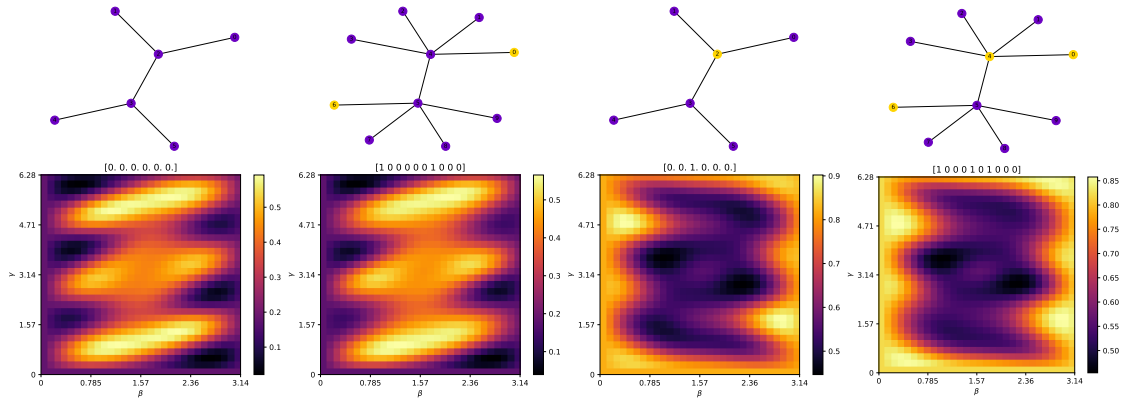

Figure 6.9: Examples where 3-regular and 5-regular landscapes look virtually identical.

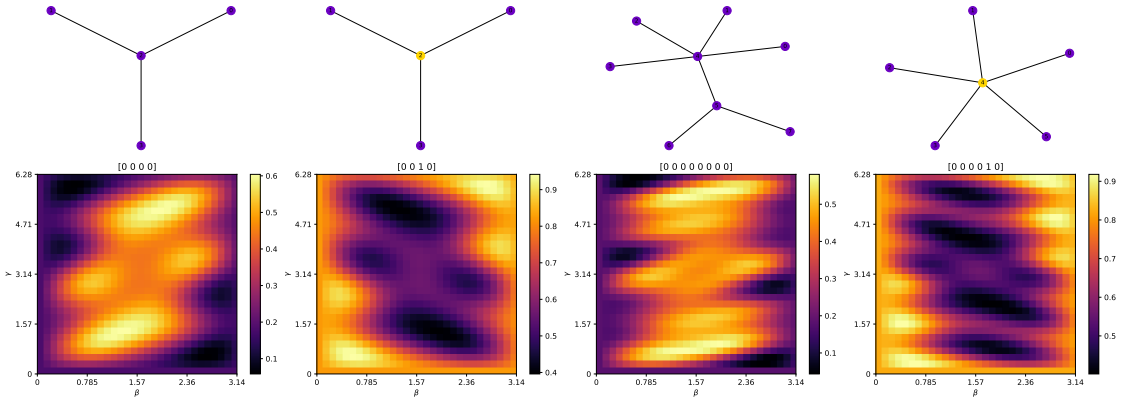

Figure 6.10: Examples of the effect of canceling warm-started nodes. Compare the landscapes on the left with Figure 6.2 on the right with Figure A.5. These landscapes indicate that there is a relationship between subgraphs of non-regular and regular graphs when warm-starting is introduced.

In a way, the two differently warm-started nodes seem to cancel each other out. This effect seems to reach beyond regular graphs. If not both central nodes have these warm-started neighbors but only one, the removal of the two opposing nodes would give a graph that is not a subgraph of a regular graph anymore. This would be beyond the domain of subgraphs of 3-regular graphs. While an analysis of non-regular graphs is out of the scope of this thesis, Figure 6.10 is further evidence for this canceling effect as it shows the resulting subgraphs. The analysis of random graphs and a concise mathematical proof of this effect is left for future works.

6.3 Transferability Map

While it is sometimes easily visible that the maxima are located close to each other, it helps to quantify this property. So instead of arguing with plots of the energy landscapes, the transferability is put into numbers. This also helps to compare similar-looking landscapes where the location of the maxima only differ slightly. The number that encapsulates transferability is the transferability

coefficient. Galda et al. [12] calculated this coefficient by optimizing 20 pairs of parameters for a donor graph and analyzed how they performed on an acceptor graph compared to a pair of parameters that were optimized on the acceptor graph directly.

The approach used here is less sophisticated. Instead of performing several optimization runs to find good parameters, good parameters are taken directly from the already calculated landscapes. At first, the landscape of the acceptor is normalized to $[0, 1]$. The location of a maximum in the donor's landscape is transferred to the landscape of the acceptor. Since the landscapes have symmetries, there might be several maxima. So for each maximum, look at the acceptor landscape and take the energy at this location. The transferability coefficient is the average of these values.

Since the landscape isn't sampled very finely (30 by 30 samples for γ and β), the parameters aren't truly optimal for the donor graph, because they could be further optimized. The same is true for the maximum energy of the acceptor. But, since the landscapes appear smooth, the resulting coefficient still gives a fairly good indication of transferability.

	3reg0	4reg0	5reg0
3reg0	-	0.500	0.953
4reg0	0.500	-	0.500
5reg0	0.941	0.500	-

Table 6.1: Transferability coefficient for the cold-started regular graphs. A row represents the acceptor graph, a column the donor.

To get a feeling for this coefficient, Table 6.1 shows the transferability coefficients for the subgraphs of 3-, 4- and 5-regular graphs, where no two nodes are merged. A value of 1 would be optimal transferability, all maxima land in a maximum. A value of 0 would mean that all maxima land at a minimum. As noted in [12] transferability is a directional property. Otherwise, the table would be completely symmetrical. As expected, the coefficient between 3reg0 and 5reg0 is high, while 4reg0 doesn't transfer as well to the other two. In fact, the coefficient of 0.5 arises because exactly half of the locations fall into a high value $(0.5 + c)$ and the other half falls into a low value $(0.5 - c)$ as can be seen when comparing Figures 6.6 and 6.7.

Another measure that can be applied is the (normalized) average difference. As the name implies, the landscapes are normalized and their average absolute difference is calculated. The normalization is done because in a landscape we only care about a value relative to its maximum/minimum. An average difference of 0 means the landscapes are identical if both are normalized. An example of this measure can be seen in Table 6.2. This measure is symmetrical. Since every value of the landscape is taken into account, it serves as a good measure to tell us how similar two landscapes are as a whole to each other.

	3reg0	4reg0	5reg0
3reg0	-	0.182	0.056
4reg0	0.182	-	0.161
5reg0	0.056	0.161	-

Table 6.2: Average energy difference for the cold-started regular graphs.

Galda et al. [12] calculated the transferability coefficient for all subgraphs of regular graphs up to degree $d \leq 8$ and visualized the resulting matrix in a transferability map. They went even further and calculated this coefficient for all possible subgraphs of random graphs up to degree $d \leq 6$. The result was a 36×36 and a 56×56 image respectively. For warm-starting, going up to degree $d = 8$ is out of the scope of this work since the number of initial cuts that need to be considered is growing too fast with increasing d. Also, random graphs aren't considered here. The analysis is performed only for the subgraphs of 3-, 4- and 5-regular graphs. Still, this constitutes a 183×183 matrix. The image showing the transferability coefficients can be seen in Figure 6.11. The average differences are depicted in Figure 6.12.

Because of this already large size, not all possible initial cuts went into the creation of this map. Left out were the ones where virtually identical landscapes existed. These are the initial cuts belonging to the tiny landscapes present for example in Figure 6.4. There is no straightforward way to order the initial cuts, the order is described in Appendix A on page 59. Also depicted in Appendix A are all initial cuts in this order, Figure A.10 on page 70 serves as a reference.

The best way to explore similarities and find patterns among the different graphs and landscapes is through an interactive version of these two maps provided in the GitHub repository [19]. Also included are instructions on the setup and usage. I highly suggest using the maps this way as a lot of the patterns described here are easier understood in this interactive manner.

The map is divided into blocks by white, dashed lines. Each block represents a graph type. A row is an acceptor, a column is a donor subgraph. Similar to the landscapes, a bright spot indicates good transferability. Oftentimes a block is divided into four smaller rectangular blocks, where two are dark and the other two are bright. Sometimes more pronounced, sometimes not as clear. This is because the graphs are ordered in such a way that all initial cuts where the two central nodes are initialized with the same value come first and then the ones where the central nodes are initialized differently. It was expected that these two groups of initial cuts have good transferability within each group, but not across them.

The coefficients between the 3- and 5-subgraphs in general appear higher than the coefficients between 4 and any of the other two. This is however not as clear-cut and there are a lot of exceptions For example, there are some 3- and 4-subgraphs that have a high transferability coefficient in both directions. 3reg1 (3) and 4reg0 (6) have coefficients of about 0.90 and 0.97.

Another interesting observation is that the 4-subgraphs seem better as acceptors than donors. The rows of the 4-subgraphs have a lot of relatively bright spots, while the columns appear darker in general.

Figure 6.12 paints a clearer picture. The map is way more smooth since not just the maxima go into the average difference, but all values of the landscape. Notice, that the colormap is inverted such that similarity, i.e. a value close to 0, is still represented by a bright spot. Because of the smoothness, some interesting patterns emerge. The big diagonal blocks, those where the degree for the central nodes are the same, for example, show bright lines that stretch across the blocks and aren't part of the main diagonal. These lines come from the fact, that each landscape for a graph where two nodes are merged, already appeared similar in the most general graph, where no two nodes are merged. The landscapes of 3reg1 are a subset of the landscapes of 3reg0 so to speak. If we take a look at the blocks where 3- and 5-subgraphs are compared, we can also see bright diagonals. These are the ones where virtually identical landscapes exist between 3- and 5-subgraphs as depicted in Figure 6.9.

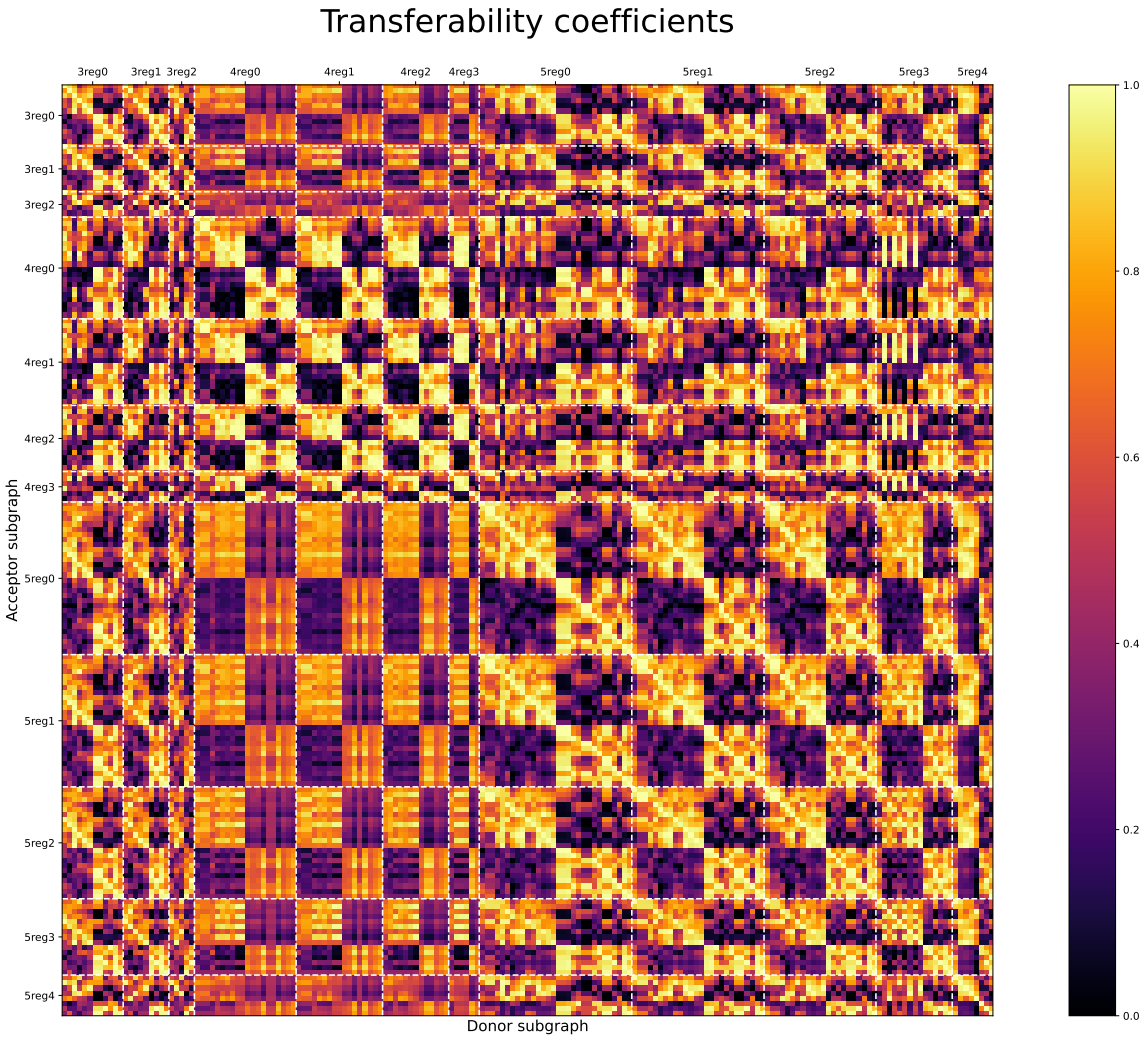

Figure 6.11: Transferability coefficients between all initial cuts and subgraphs up to degree $d \leq 5$. The acceptor graph is given by the row, the donor by the column. The graph types are separated by white dashed lines. For the order within a block refer to Figure A.10 on page 70.

6.4 Parameter Concentration

With the knowledge of the influence of subgraphs, we can shift the focus to larger instances. The goal is to see whether or not the insights gained on the subgraphs apply in a more relevant case. Similar to [5], I investigate if the WS-QAOA parameters concentrate for similar instances. This question is answered by transferring parameters between instances and comparing their performances.

This numerical experiment is constructed as follows: First, create a random d-regular graph with 20 nodes, the *donor* graph. Here, I examine 3-, 4- and 5-regular graphs. Next, calculate a good initial cut for this instance by generating an approximate cut with the GW algorithm. The key step is then to transfer pairs of parameters $(\boldsymbol{\gamma}, \boldsymbol{\beta})$ from this instance to other random d-regular instances. In total, there are 30 instances where the performance of the parameter pairs is evaluated. For each

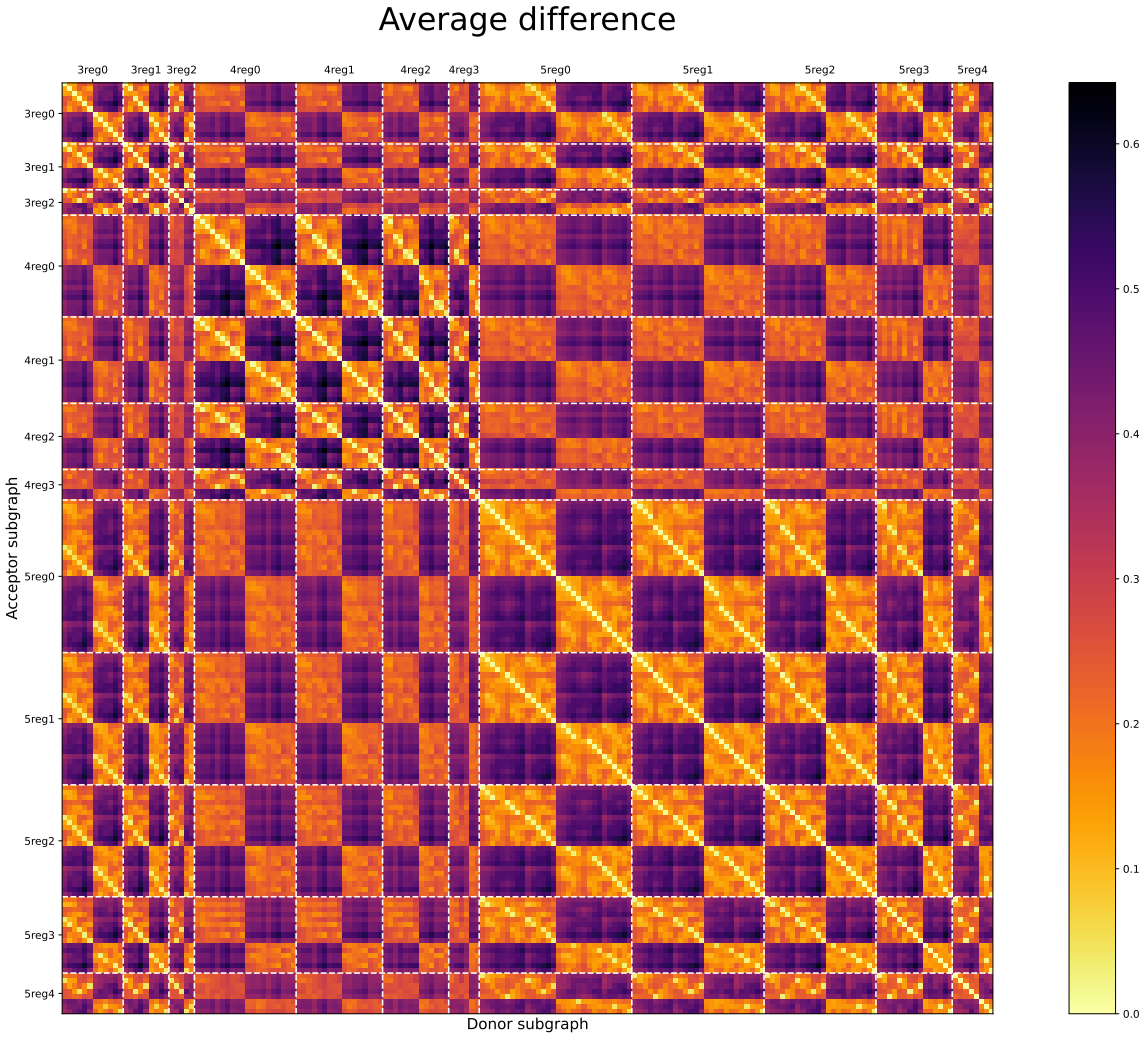

Figure 6.12: Normalized average difference between all initial cuts and subgraphs. To signify similarity as bright, the colormap is inverted, such that 0 is bright and higher values are darker. The acceptor graph is given by the row, the donor by the column. The graph types are separated by white dashed lines. For the order within a block refer to Figure A.10 on page 70.

instance, the GW algorithm is run again to determine a good cut for the new random instance. This cut is then used as the initial cut for this instance. As was done by Brandao et al. [5], there are three groups of parameters that are tested for each instance. Parameters that are optimized to give a *Low* objective function, *Random* parameters and parameters that are optimized to give a *High* objective function value. How exactly these parameters are determined is explained in the next paragraph. To get the energy, a simulation is run with the parameters where the depth is fixed at $p = 1$. The performance is measured as the coefficient of energy obtained by these parameters, divided by the true MaxCut value, the coefficient is further called the approximation ratio. The result of these experiments in the form of the mean and standard deviation of the 30 coefficients is shown in Table 6.3.

To obtain parameters that result in a high objective value the warm-started donor graph is decomposed into its subgraphs. With these subgraphs, an energy landscape can be constructed, by summing the corresponding landscapes weighted by how often the respective subgraph appears in the donor. The location with the highest energy is then chosen as an initial point for further optimization, which runs until the value converges. The parameters for a low objective function value are obtained in a similar manner, as an initial point, the location with the lowest energy is chosen. In the optimization step, the goal is to achieve low energy. For the group of random parameters, the parameters are chosen uniformly at random for each instance. Further details on what graph instances and what parameters were used can be found in Appendix B on page 71.

	Low		Random		High	
	Mean	Std.	Mean	Std.	Mean	Std.
3-regular	0.483	0.016	0.719	0.121	0.933	0.036
4-regular	0.534	0.022	0.748	0.131	0.928	0.041
5-regular	0.563	0.021	0.749	0.098	0.932	0.038

Table 6.3: Mean approximation ratios and their standard deviation for transferring parameters of low energy, random parameters or high energy. 30 random regular graphs with 20 nodes each were used. Each graph energy was calculated with a depth of $p = 1$.

The results of Table 6.3 are rather surprising. Every graph type had a similar result in each parameter category. For bad parameters, the approximation ratio is around 0.50, where the higher the degree, the higher the ratio. Further investigations are necessary to see if this pattern continues with increasing d. Even the random instances had a ratio of about 0.72 and upwards. An unexpected high performance was achieved with the parameters optimized to give a high objective function value. The approximation ratio over all instances was about 0.93. For the different d-regular donors, with $d \in \{3, 4, 5\}$, the approximation ratio was at 0.930, 0.969 and 0.924 respectively, meaning some instances outperformed the donor in terms of approximation ratio.

This result comes as a surprise as the previous findings suggest, that parameter transferability among warm-started graphs isn't as straightforward as in the cold-start case. Nevertheless, the resulting ratios are very high and the transferred parameters perform about the same as for their donors. To explain why the results are this high we have to take a closer look at the donors. Figure 6.13 depicts the donor graphs and their corresponding energy landscapes. The graphs are shown with their approximate cut, a red node is initialized with 1, and a blue node with 0. Edges part of the cut are colored in orange.

The landscape of the 3-regular donor gives a hint to the reason for this good transferability. Going back to the landscapes for the subgraphs of regular graphs, one can see that this donor landscape shares a great resemblance with the landscape of 3reg0 (6) and 3reg0 (7) or rather the corresponding graph where two nodes are merged, 3reg1 (5) (cf. top right of Figure 6.2). In fact, from the 30 total subgraphs of the donor, these graphs appeared 7, 10 and 6 times respectively. This is the reason for this similar-looking landscape. Why these subgraphs appear so frequently can also be explained. These subgraphs all represent cuts with a high value. The first one is the optimal subgraph in the sense that all its 6 edges are in the cut. The other two have a cut value of 5 and the third subgraph is also the optimal one for the 3reg1 category.

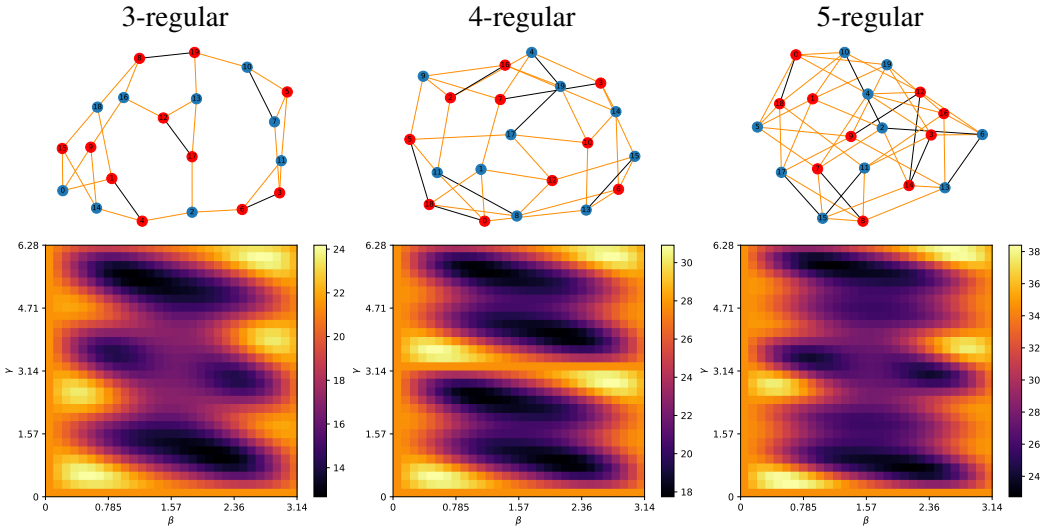

Figure 6.13: The donor graphs with an approximate cut as warm-starting. Edges that are part of the cut are highlighted in orange. Below each graph is the landscape that was used to find good/bad initial points.

For the 4-regular variant, the same observations can be made. 10 of the 40 subgraphs are isomorphic to 4reg0 (11) whose landscape most closely resembles the one of the donor graph (cf. Figure A.1 bottom row, second from the left). A further 6 are the 4reg1 equivalent (4reg1 (9)). Similarly for the 5-regular donor: The most frequent subgraph of the 5-regular donor is 5reg0 (16) (see Figure A.5 top row, second from the right).

It is also worth taking a closer look at the high approximation ratios when transferring good parameters. For the 3-regular version, the acceptors had on average the same ratio as the donor. Some even performed better than the donor, as can be seen in Table B.1 on page 72. This ratio however depended strongly on the quality of the initial cut. Instances where the approximate cut was simultaneously the MaxCut, had a higher ratio than the donor itself. Conversely, approximations where the cut had a value far below the maximum also resulted in a ratio that was below the donor's. Nevertheless, these examples demonstrate that, between instances, good parameters lie near to each other and that the transfer of parameters between similar instances can be successful.

6.5 Landscape Approximation

The findings of the previous section, together with the insights into the similarities between landscapes, lead to ideas that may improve the execution of WS-QAOA. The goal is to find good initial points for the parameters and save optimization steps for the depth $p = 1$. Similar to the procedure described in Section 5.2, the landscape of a warm-started 3-regular graph can also be precomputed. But compared to the cold-started case, one has to precompute 27 different landscapes instead of 3. Calculating 27 different landscapes in advance is still feasible, nevertheless one can try to decrease this number. For example, we can try to leverage the similarities between the landscapes so that we only have to consider certain subgraphs. As described in Section 6.2, the landscapes of 3reg0 contain all the relevant landscapes that appear for 3-regular graphs. 3reg0 is the most

general graph of the 3 different subgraph types in the sense that it allows for the most possible different initial cuts (12 in total). A different type of subgraph, such as 3reg1 or 3reg2, can be treated as its 3reg0 counterpart. The counterpart is determined by splitting the relevant nodes (see Section 6.2).

Another observation we can incorporate is that the most frequent subgraph has a big influence on the resulting landscape. Take the 4-regular donor of the previous section as an example. 16 of the 40 total subgraphs had a landscape that resembled the one of the donor. These 16 subgraphs could be divided into two isomorphic groups, i.e. these were the two most frequent types of subgraphs. For the purposes here, this translates to only considering the most frequent subgraphs until a certain portion of the edges is covered, for example, $\frac{16}{40} = 0.4$. The result of these shortcuts is best demonstrated by an example.

Figure 6.14 shows two landscapes of a warm-started, random 4-regular graph with 20 nodes. For details refer to Appendix B on page 71. The approximate one was created by only using the two most frequent subgraphs in the calculation, the other shows the true landscape. These accounted for 20 of the 40 total subgraphs.

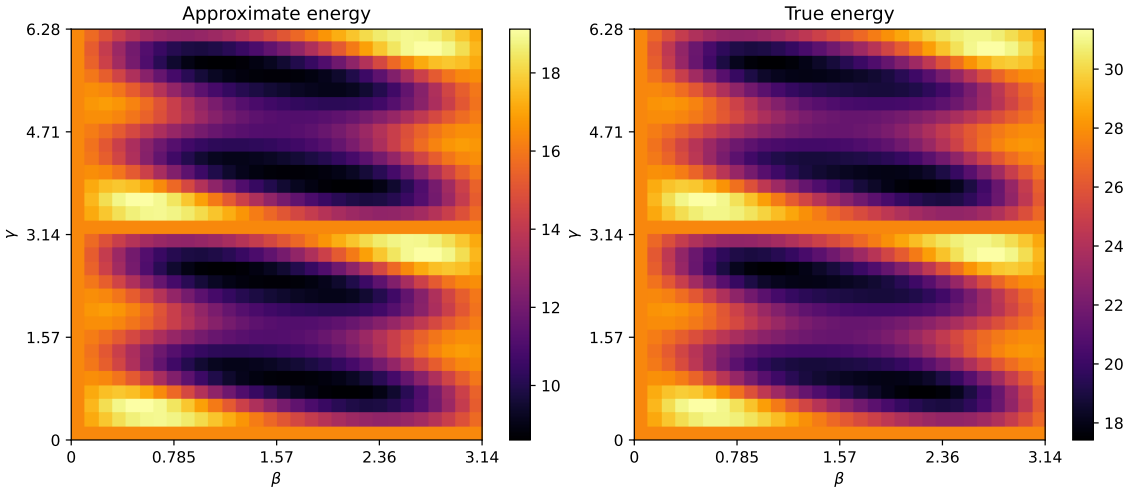

Figure 6.14: Landscapes of a warm-started 4-regular graph with 20 nodes. The left landscape only used the two most frequent subgraphs for calculation. Because of this, the range of the values is not as big as the true landscape.

The similarity between the landscapes is clearly visible. The landscapes also closely resemble the 4-regular one of Figure 6.13, which is further evidence for the parameter concentration. Apart from being similar looking, the location of the maxima is again almost the same. Each landscape has 30×30 samples and 2 of the 4 maxima are at the exact same location. The other two are offset by one sample in the β-direction, which is $\frac{\pi}{30} \approx 0.105$. If the landscapes were finer sampled a measurable offset is expected, nevertheless the maxima are located close to each other.

This example demonstrates the potential of the approach. But, one doesn't even have to calculate the landscape to get usable parameters. Instead, a subgraph optimization can be performed like in Section 5.2. The advantage is that only small graphs need to be considered which might be better suited for the hardware, classical simulators or quantum. Galda et al. [12] had a related approach where they took one small donor graph and optimized it to transfer the parameters to larger instances.

In the example here, optimizing the parameters over all subgraphs gave an approximation ratio of 0.92850. Only considering the two most frequent subgraphs of this instance lead to an approximation ratio of 0.92827. The first optimization took 57 seconds on commodity hardware while the second one took 10 seconds. Both optimizations were run in Qiskit's `'statevector_simulator'` which may not be the most efficient piece of software to calculate the energies.

There are some caveats to this technique. Because the regular graphs have only a few possible subgraphs, there are a lot of warm-started subgraphs that are isomorphic to each other especially when the graph is initialized with a good approximation. This is likely to change when more general graphs are considered such as random graphs. The many types of subgraphs make it less likely that multiple subgraphs are isomorphic to each other when warm-starting is introduced. This is a topic for future works.

7 Discussion

This chapter discusses conjectures concerning the transferability posed by other works and whether or not they still hold for the warm-started case.

As described in Section 4.1, there are three sufficient conditions that, if fulfilled, allow for a transfer of optimal parameters [12]. These conditions made no restrictions on the graphs and thus they also hold in the case where warm-starting is used. It is however really unlikely if not impossible to fulfill all these conditions in conjunction with warm-starting. To reiterate, all subgraphs within a donor graph should be mutually transferable. If an approximate MaxCut is used as a warm-start, it almost certainly has an edge, where the incident nodes have the same initialization. Only if all edges are part of the cut this doesn't happen. As discussed in the previous section these two types of subgraphs (central edge part of the cut or not) show poor transferability in general. The conditions are therefore too strict to be applied in the warm-start case.

Additionally, Galda et al. [12] proposed three conjectures that stated if a parameter transfer produces good results. While they left the proof of these conjectures to future work, it can be stated whether or not these conjectures still hold for d-regular graphs with $d \leq 5$ if warm-starting is introduced. The 3 conjectures were:

1. Optimized parameters can be successfully transferred between any random d-regular graphs, $d \in \mathbb{N}$.
2. Optimized parameters can be successfully transferred from any random d_1-regular graph to any random d_2-regular graph, assuming that d_1 and d_2 are either both odd or both even.
3. Optimized parameters cannot be successfully transferred between d_1- and d_2-regular graphs, if d_1 is odd and d_2 is even, or vice versa.

To examine these conjectures, I argue with the landscapes of the subgraphs. More specifically, the transferability map shown in Figure 6.11 serves as an excellent tool to help answer the question. For the first conjecture it can be safely said, that in general not all parameters are transferable between d-regular graphs when warm-starting is employed. The reason is that we can divide the landscapes into two groups, one where the central edge is part of the cut and one where it isn't. As described in Section 6.2, the landscapes of these groups look vastly different from each other and sometimes parameters that are optimal for one graph are suboptimal for the other. In the transferability map this can be seen by the division of each block into 4 smaller blocks.

The second conjecture is also refuted by the same argument, in general warm-starting does not permit an easy transfer between 3- and 5-regular graphs. However there are a lot of instances where a successful transfer is possible. Recall the instances where the landscapes were virtually identical between 3- and 5-subgraphs (Figure 6.9). This effect didn't appear between the 4-subgraphs and any of the other two types. Again looking at the transferability map, the 5-subgraphs seem to accept

the parameters of the 3-subgraphs on average better that the 4-subgraphs. The difference map also suggests that the 3-subgraphs are more similar to the 5-subgraphs and vice versa, than to the 4-subgraphs as the corresponding blocks appear brighter.

The third conjecture holds in the most cases. There are, however, exceptions to this. For example, take the two graphs 3reg1 (3) and 4reg0 (6). Transferring from 3reg1 (3) to 4reg0 (6) yields a coefficient of 0.969 and the other way around we get a coefficient of 0.895. It should be noted, that these high coefficients aren't that frequent between those two categories.

In conclusion, when using warm-starting the first conjecture alone can't be taken as a basis to justify successful parameter transfer. One important observation is that the central edge is of great importance for the transfer. To increase the probability of success the acceptor- and donor-subgraph both should have a central edge that is part of the cut or not. And while the other two conjectures often times hold true, there are exceptions that don't fit these rules. In general one has to be careful when using warm-starting and transferring parameters.

8 Conclusion and Outlook

Transferring parameters between warm-started QAOA instances isn't as easily justified as for the cold-started variant. This is mainly because there are many different initial cuts of the same subgraphs that have a low transferability coefficient. Nevertheless, a transfer can succeed. This thesis provides assistance for choosing appropriate instances to facilitate a transfer.

To reduce the complexity of the problem, the hyperparameters were fixed at $\varepsilon = 0.1$ and $p = 1$. As a simple case, I first analyzed 3-regular graphs. There were stark differences in the landscapes depending on the initial cut. The different subgraph types, however, showed similar landscapes when both types had corresponding initial cuts. Between those, parameters are transferable. For the 4- and 5-regular versions similar patterns appeared. More interestingly, the landscapes of the 3-subgraphs appeared slightly altered among the 5-subgraphs. Transferability from 4-subgraphs to the other two is, in general, less likely to be successful. When transferring parameters between subgraphs, the central edge for both the donor and acceptor should be part of the cut or both shouldn't.

The presented transferability map serves as a tool to find good donor subgraphs and as a summary of the patterns described in this thesis. It can also help to identify new patterns that aren't described in this thesis. Further, I demonstrated with numerical experiments, that 3-, 4- and 5-regular graphs exhibit a concentration of good parameters. The explanation of the experiments also gave a reason on why parameters can be transferred in the warm-started case even if the landscapes of subgraphs differ significantly.

From these insights, I demonstrated my approach to precomputing parameters with *landscape approximation*. It seems very promising for the regular graphs analyzed in this thesis, especially in conjunction with the results on parameter concentration. For the random graphs, this approach might be significantly less potent. This needs to be confirmed with further research.

When warm-starting is introduced, the conjectures of Galda et al. [12] don't hold in general. However, conjecture 3: *Optimized parameters cannot be successfully transferred between d_1- and d_2-regular graphs, if d_1 is odd and d_2 is even, or vice versa* seems to hold in most cases.

A big problem that comes with WS-QAOA for MaxCut is the increased complexity. The number of cases that need to be considered grows exponentially as the number of nodes increases. For this reason, the scope of this thesis is limited in the sense that the results mainly serve as an entry point into precomputing parameters for WS-QAOA. The application of the results in practice remains limited as unweighted regular graphs are very restrictive and don't represent graphs that are of high practical interest. At the same time, the approaches aren't ruled out.

In future work, random graphs or weighted MaxCut should be considered. Especially random graphs seem promising as a relationship to the landscapes of regular graphs is conjectured. There has been research into random graphs for the cold-start case including the main research this thesis is based upon [12]. Parameter transfer for weighted MaxCut has also been already considered in the cold-start version [22].

So far, ε was fixed at $\varepsilon = 0.1$. If the same conclusions can be made for different ε is to be determined. Similarly, in this thesis, the depth was fixed at $p = 1$. This was justified by the advantages of WS-QAOA but better performance is expected with increasing p. Also, the version of WS-QAOA investigated here used a mixer whose ground state is the initial state. Egger et al. [10] proposed a modified mixer that had a good performance at $\varepsilon = 0.25$. It would be interesting to see if and how the results differ with this modified mixer.

Since the class of VQAs is a promising candidate for NISQ devices it would be of value to have similar initialization methods for other algorithms. There might be parallels to QAOA for MaxCut.

Bibliography

[1] V. Akshay, D. Rabinovich, E. Campos, J. Biamonte. "Parameter concentrations in quantum approximate optimization". In: *Physical Review A* 104.1 (July 2021). ISSN: 2469-9934. DOI: 10.1103/physreva.104.l010401. URL: http://dx.doi.org/10.1103/PhysRevA.104.L010401 (cit. on pp. 23, 25).

[2] M. Beisel, J. Barzen, F. Leymann, F. Truger, B. Weder, V. Yussupov. "Patterns for Quantum Error Handling". In: *Proceedings of the 14th International Conference on Pervasive Patterns and Applications (PATTERNS 2022)*. Xpert Publishing Services (XPS), Apr. 2022, pp. 22–30. ISBN: 978-1-61208-953-9 (cit. on p. 23).

[3] M. Born, V. Fock. "Beweis des Adiabatensatzes". In: *Zeitschrift fuer Physik* 51.3-4 (Mar. 1928), pp. 165–180. DOI: 10.1007/bf01343193. URL: https://doi.org/10.1007/bf01343193 (cit. on p. 18).

[4] S. Boulebnane, A. Montanaro. *Predicting parameters for the Quantum Approximate Optimization Algorithm for MAX-CUT from the infinite-size limit*. 2021. DOI: 10.48550/ARXIV.2110.10685. URL: https://arxiv.org/abs/2110.10685 (cit. on p. 25).

[5] F. G. S. L. Brandao, M. Broughton, E. Farhi, S. Gutmann, H. Neven. *For Fixed Control Parameters the Quantum Approximate Optimization Algorithm's Objective Function Value Concentrates for Typical Instances*. 2018. DOI: 10.48550/ARXIV.1812.04170. URL: https://arxiv.org/abs/1812.04170 (cit. on pp. 25, 26, 44, 45).

[6] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, P. J. Coles. "Variational quantum algorithms". In: *Nature Reviews Physics* 3.9 (Aug. 2021), pp. 625–644. DOI: 10.1038/s42254-021-00348-9. URL: https://doi.org/10.1038/s42254-021-00348-9 (cit. on p. 13).

[7] A. M. Dalzell, A. W. Harrow, D. E. Koh, R. L. L. Placa. "How many qubits are needed for quantum computational supremacy?" In: *Quantum* 4 (May 2020), p. 264. DOI: 10.22331/q-2020-05-11-264. URL: https://doi.org/10.22331/q-2020-05-11-264 (cit. on p. 15).

[8] J. Dborin, F. Barratt, V. Wimalaweera, L. Wright, A. G. Green. *Matrix Product State Pre-Training for Quantum Machine Learning*. 2021. DOI: 10.48550/ARXIV.2106.05742. URL: https://arxiv.org/abs/2106.05742 (cit. on p. 25).

[9] S. J. Devitt, W. J. Munro, K. Nemoto. "Quantum error correction for beginners". In: *Reports on Progress in Physics* 76.7 (June 2013), p. 076001. DOI: 10.1088/0034-4885/76/7/076001. URL: https://doi.org/10.1088/0034-4885/76/7/076001 (cit. on pp. 13, 23).

[10] D. J. Egger, J. Mareček, S. Woerner. "Warm-starting quantum optimization". In: *Quantum* 5 (June 2021), p. 479. DOI: 10.22331/q-2021-06-17-479. URL: https://doi.org/10.22331/q-2021-06-17-479 (cit. on pp. 14, 15, 19, 22, 54).

[11] E. Farhi, J. Goldstone, S. Gutmann. *A Quantum Approximate Optimization Algorithm*. 2014. DOI: 10.48550/ARXIV.1411.4028. URL: https://arxiv.org/abs/1411.4028 (cit. on pp. 15–17, 19, 23).

[12] A. Galda, X. Liu, D. Lykov, Y. Alexeev, I. Safro. *Transferability of optimal QAOA parameters between random graphs*. 2021. DOI: 10.48550/ARXIV.2106.07531. URL: https://arxiv.org/abs/2106.07531 (cit. on pp. 19, 23–25, 27, 33, 42, 43, 48, 51, 53, 54).

[13] N. Gould, P. Toint. *A Quadratic Programming Page*. URL: https://www.numerical.rl.ac.uk/people/nimg/qp/qp.html (visited on 06/16/2022) (cit. on p. 20).

[14] E. K. Grant, T. S. Humble. *Adiabatic Quantum Computing and Quantum Annealing*. July 2020. DOI: 10.1093/acrefore/9780190871994.013.32. URL: https://oxfordre.com/physics/view/10.1093/acrefore/9780190871994.001.0001/acrefore-9780190871994-e-32 (cit. on p. 22).

[15] A. Hagberg, P. Swart, D. S Chult. *Exploring network structure, dynamics, and function using NetworkX*. Tech. rep. Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008 (cit. on p. 71).

[16] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, T. E. Oliphant. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2 (cit. on p. 71).

[17] A. W. Harrow, A. Hassidim, S. Lloyd. "Quantum Algorithm for Linear Systems of Equations". In: *Physical Review Letters* 103.15 (Oct. 2009). DOI: 10.1103/physrevlett.103.150502. URL: https://doi.org/10.1103/physrevlett.103.150502 (cit. on p. 23).

[18] N. Jain, B. Coyle, E. Kashefi, N. Kumar. *Graph neural network initialisation of quantum approximate optimisation*. 2021. DOI: 10.48550/ARXIV.2111.03016. URL: https://arxiv.org/abs/2111.03016 (cit. on p. 25).

[19] J. Obst. *GitHub repository*. 2022. URL: https://github.com/obstjn/ws-qaoa-transfer (visited on 07/20/2022) (cit. on pp. 43, 71).

[20] A. Papachristodoulou, J. Anderson, G. Valmorbida, S. Prajna, P. Seiler, P. A. Parrilo, M. M. Peet, D. Jagt. *SOSTOOLS: Sum of squares optimization toolbox for MATLAB*. Available from https://github.com/oxfordcontrol/SOSTOOLS. 2021. URL: http://arxiv.org/abs/1310.4716 (cit. on p. 20).

[21] S. Poljak, F. Rendl, H. Wolkowicz. "A recipe for semidefinite relaxation for (0, 1)-quadratic programming". In: *Journal of Global Optimization* 7.1 (July 1995), pp. 51–73. DOI: 10.1007/bf01100205. URL: https://doi.org/10.1007/bf01100205 (cit. on p. 20).

[22] R. Shaydulin, P. C. Lotshaw, J. Larson, J. Ostrowski, T. S. Humble. *Parameter Transfer for Quantum Approximate Optimization of Weighted MaxCut*. 2022. DOI: 10.48550/ARXIV.2201.11785. URL: https://arxiv.org/abs/2201.11785 (cit. on pp. 25, 54).

[23] P. W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". In: *SIAM Journal on Computing* 26.5 (Oct. 1997), pp. 1484–1509. DOI: 10.1137/s0097539795293172. URL: https://doi.org/10.1137/s0097539795293172 (cit. on pp. 13, 23).

[24] M. Stęchły. *Quantum Approximate Optimization Algorithm explained*. 2020. URL: https://www.mustythoughts.com/quantum-approximate-optimization-algorithm-explained (visited on 05/16/2022) (cit. on p. 15).

[25] D. Steurer. *Reduction from 3 SAT to MAX CUT*. Cornell University - Course CS 4821-Spring'14. Mar. 2014. URL: http://www.cs.cornell.edu/courses/cs4820/2014sp/notes/reduction-maxcut.pdf (visited on 06/17/2022) (cit. on p. 15).

[26] M. Treinish, J. Gambetta, P. Nation, qiskit-bot, P. Kassebaum, D. M. Rodríguez, S. de la Puente González, S. Hu, K. Krsulich, J. Lishman, J. Garrison, L. Zdanski, J. Yu, M. Marques, J. Gacon, D. McKay, J. Gomez, L. Capelluto, Travis-S-IBM, A. Panigrahi, lerongil, R. I. Rahman, S. Wood, L. Bello, T. Itoko, C. J. Wood, D. Singh, Drew, E. Arbel, Glen. *Qiskit/qiskit: Qiskit 0.37.1*. Version 0.37.1. July 2022. DOI: 10.5281/zenodo.6924865. URL: https://doi.org/10.5281/zenodo.6924865 (cit. on pp. 28, 29).

[27] F. Truger, M. Beisel, J. Barzen, F. Leymann, V. Yussupov. "Selection and Optimization of Hyperparameters in Warm-Started Quantum Optimization for the MaxCut Problem". In: *Electronics* 11.7 (2022). ISSN: 2079-9292. DOI: 10.3390/electronics11071033. URL: https://www.mdpi.com/2079-9292/11/7/1033 (cit. on p. 27).

[28] J. Weidenfeller. *Introduction to the Quantum Approximate Optimization Algorithm and Applications*. 2021. URL: https://learn.qiskit.org/summer-school/2021/lec5-2-introduction-quantum-approximate-optimization-algorithm-applications (visited on 05/16/2022) (cit. on p. 15).

All links were last followed on August 02, 2022.

A Supplementary Figures

Listed here, are additional figures where each is a large collection of smaller plots. For this reason, they are meant to be viewed on a computer screen where one can zoom in. The tiny landscapes represent the effect of virtually identical landscapes as described in Section 6.2

The initial cuts of the maps follow a certain order. First, consider only those initial cuts where the central edge isn't cut. It starts with the initial cut, where every node is initialized with 0. What follows are the initial cuts where the neighbors of one central node are initialized successively with 1. After that come the initial cuts where both central nodes are initialized with 1. Finally, come the initial cuts where the neighbors of the opposite node are successively initialized with 1. The initial cuts where the central edge is part of the cut are in the same order as their mirrored counterparts.

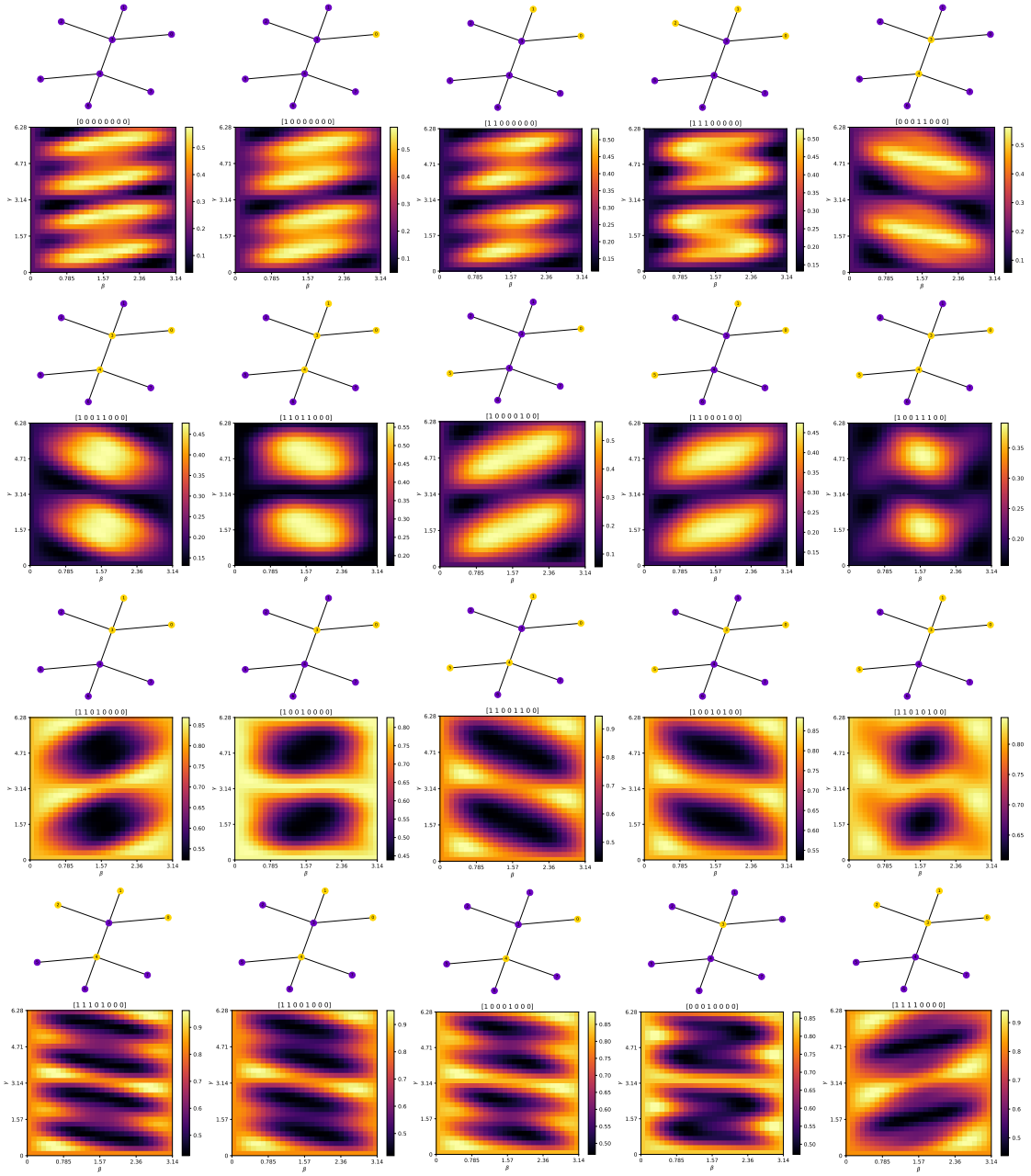

Figure A.1: All different initial cuts for graph 4reg0. The mirrored relationship between the initial cuts is visible as a reflection across the x-axis.

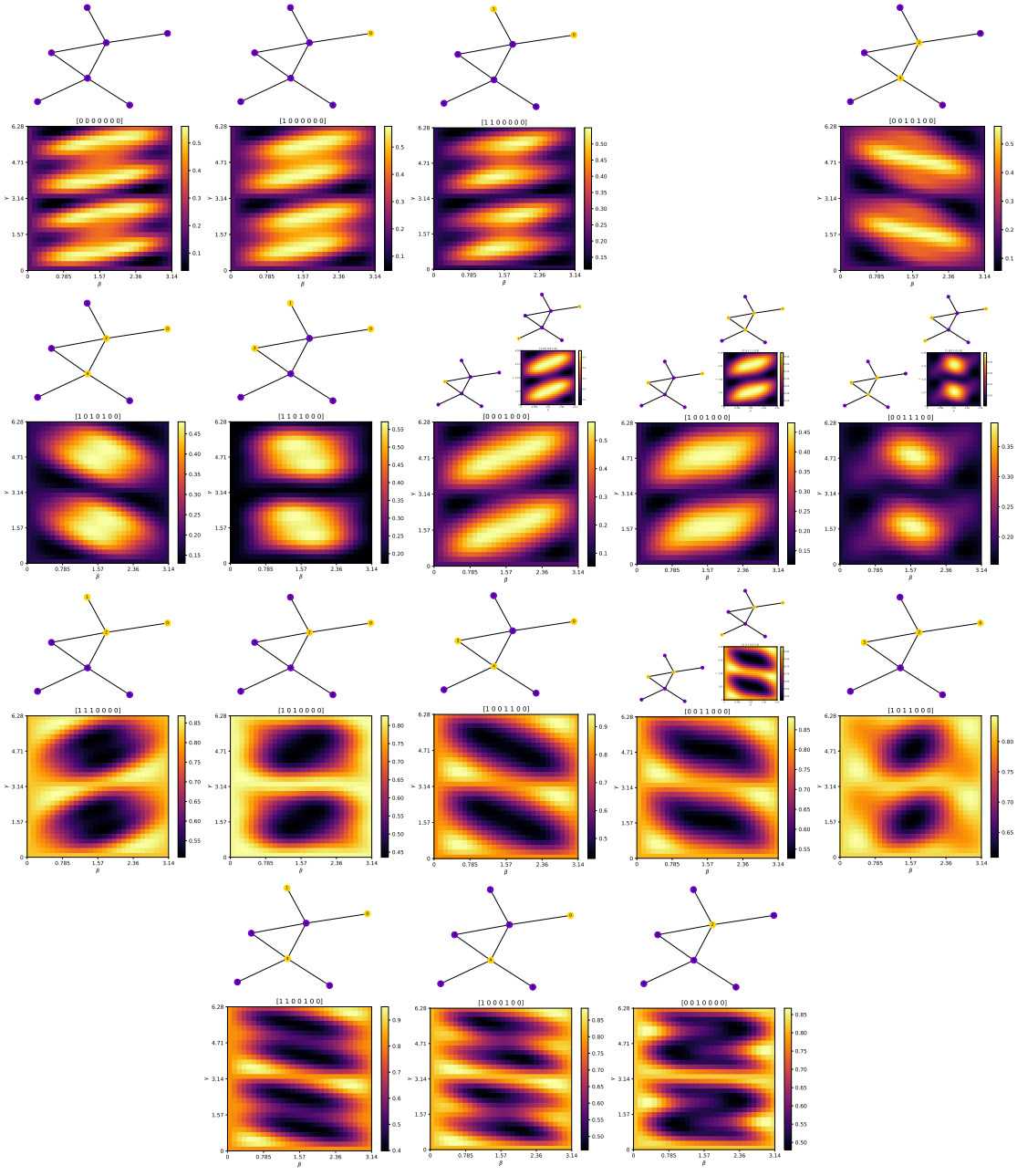

Figure A.2: All different initial cuts for graph 4reg1. The ordering is the same as in Figure A.1.

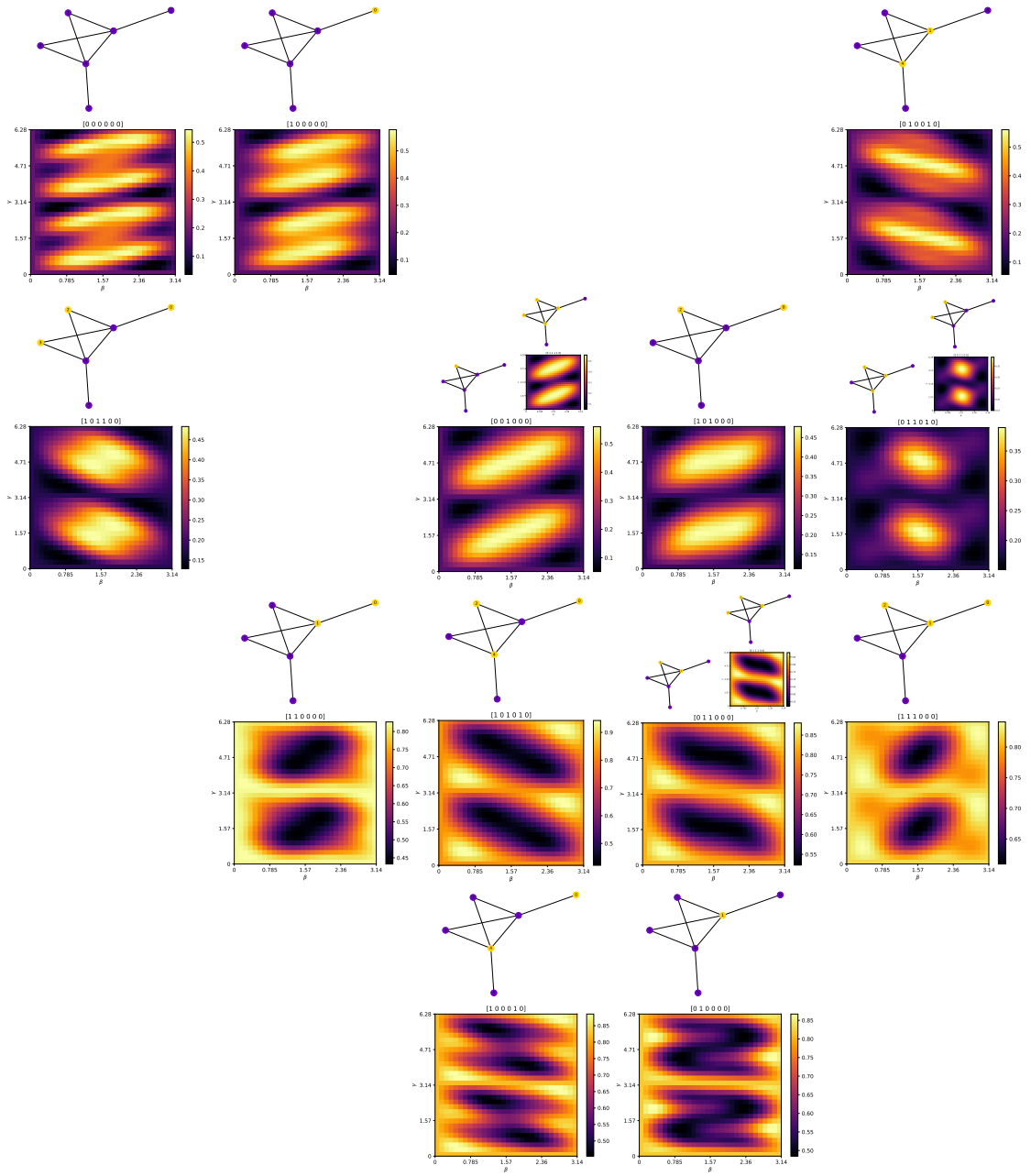

Figure A.3: All different initial cuts for graph 4reg2. The ordering is the same as in Figure A.1.

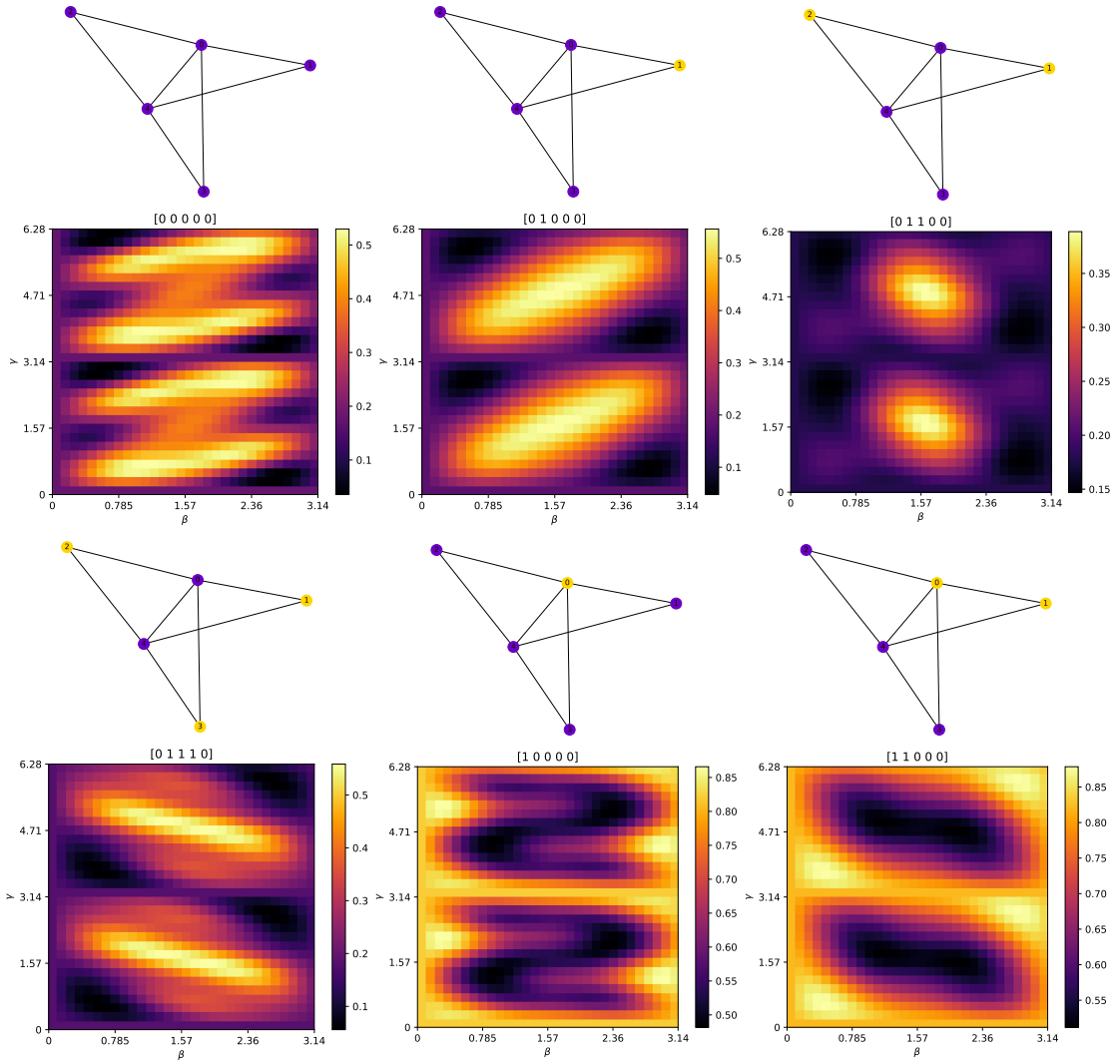

Figure A.4: All different initial cuts for graph 4reg3.

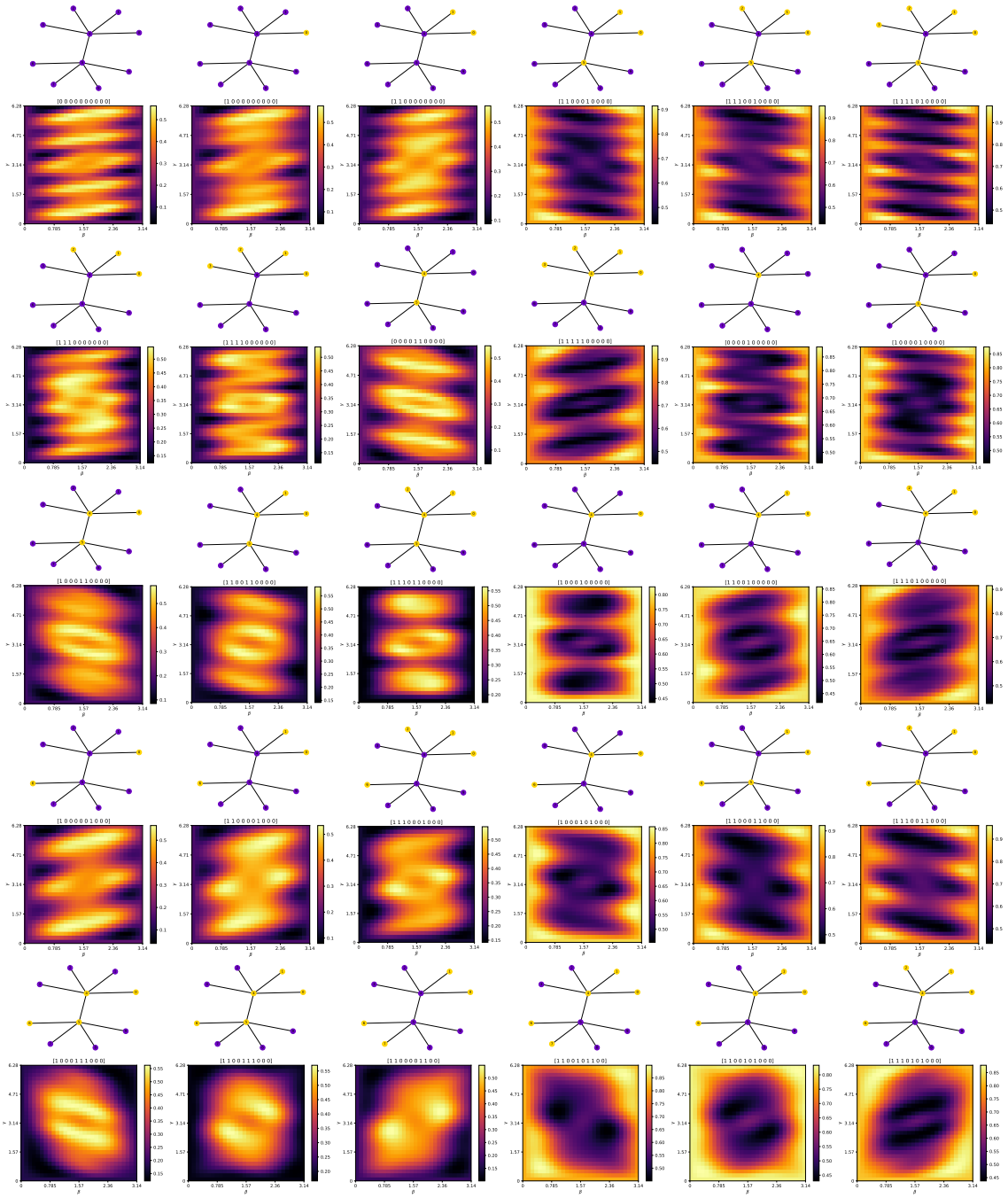

Figure A.5: All different initial cuts for graph 5reg0. They are ordered such that the mirrored relationship between the initial cuts is visible as a reflection across the x-axis.

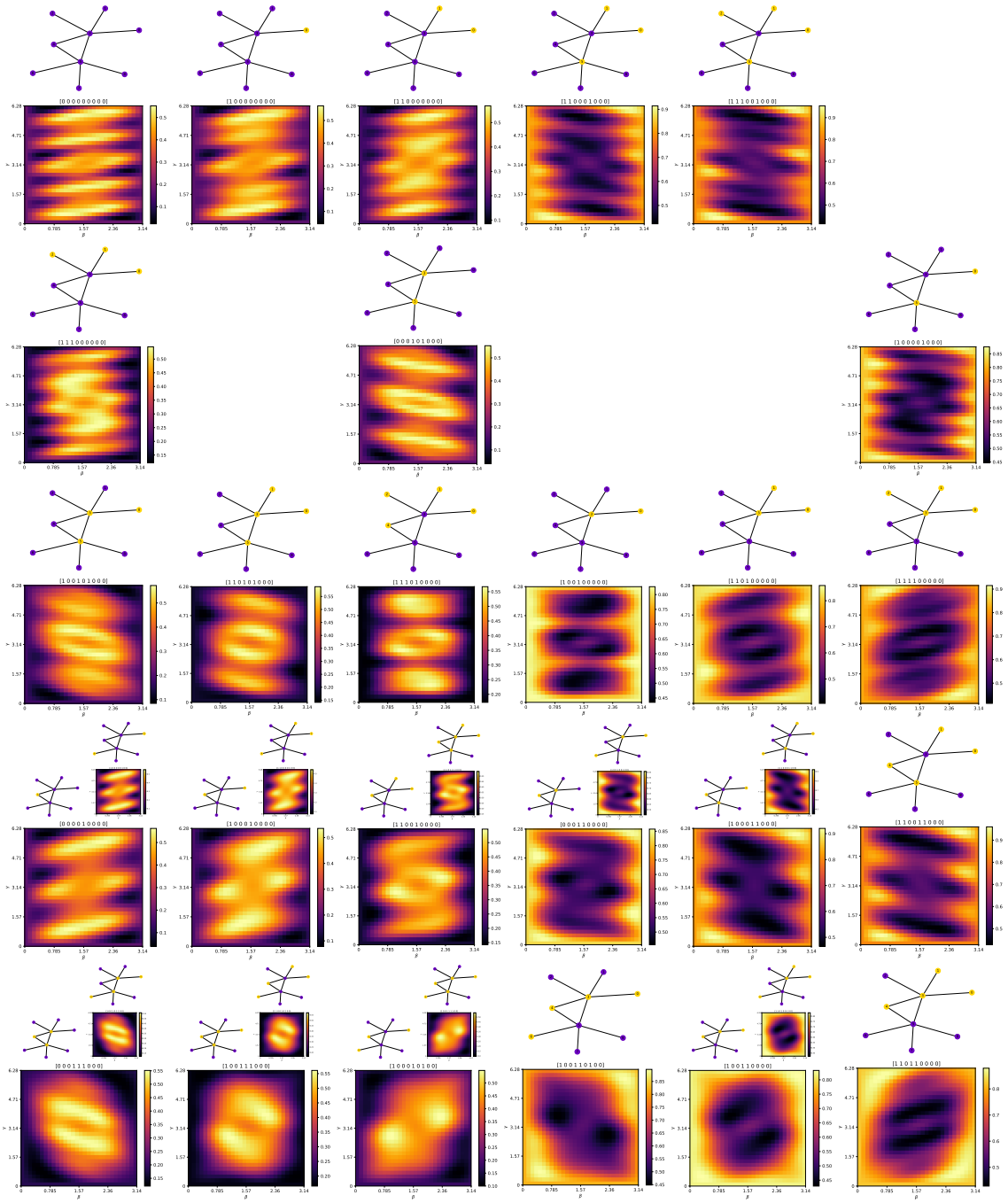

Figure A.6: All different initial cuts for graph 5reg1. The ordering is the same as in Figure A.5.

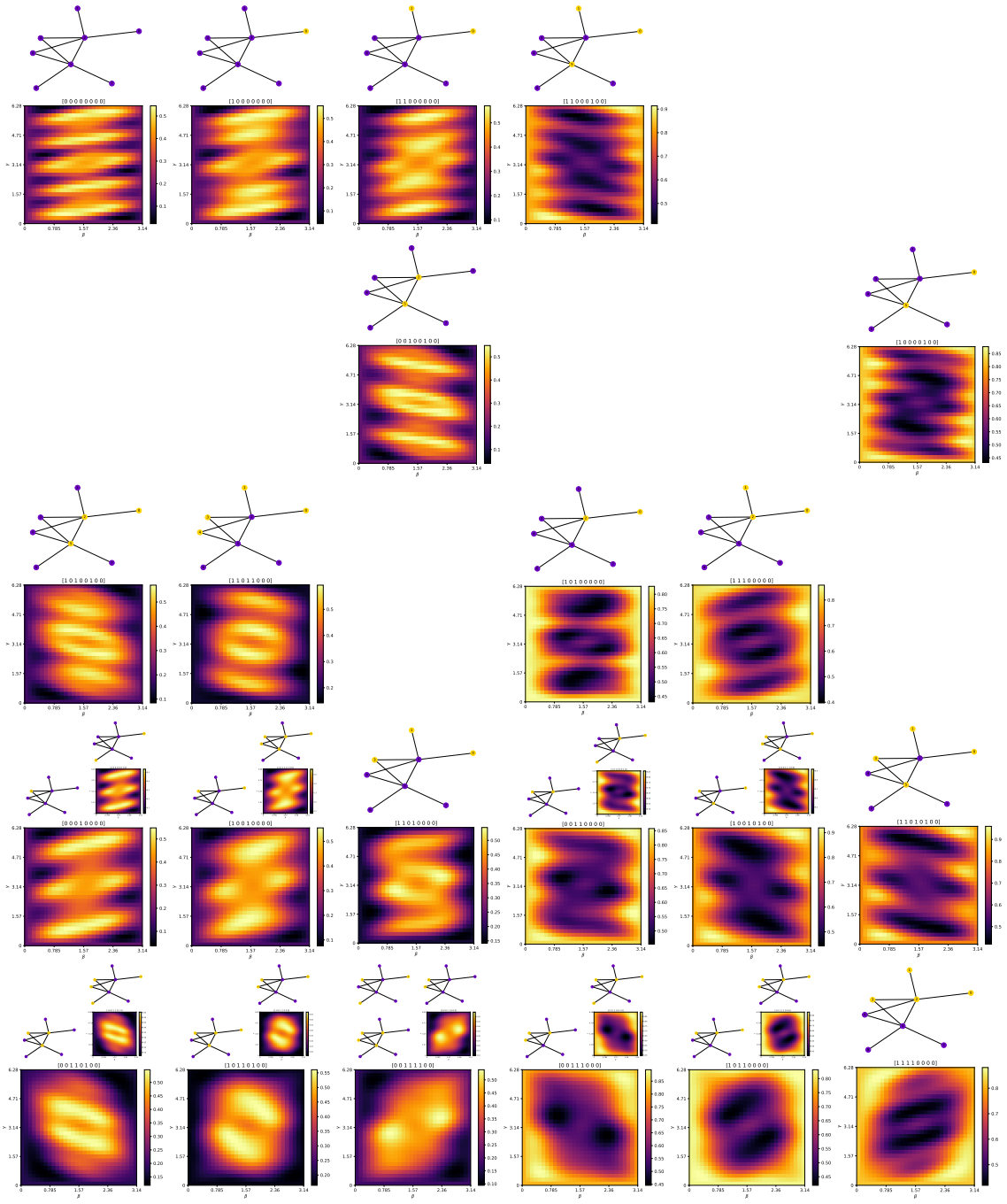

Figure A.7: All different initial cuts for graph 5reg2. The ordering is the same as in Figure A.5. The bottom row has a case, where 3 graphs have virtually the same landscape.

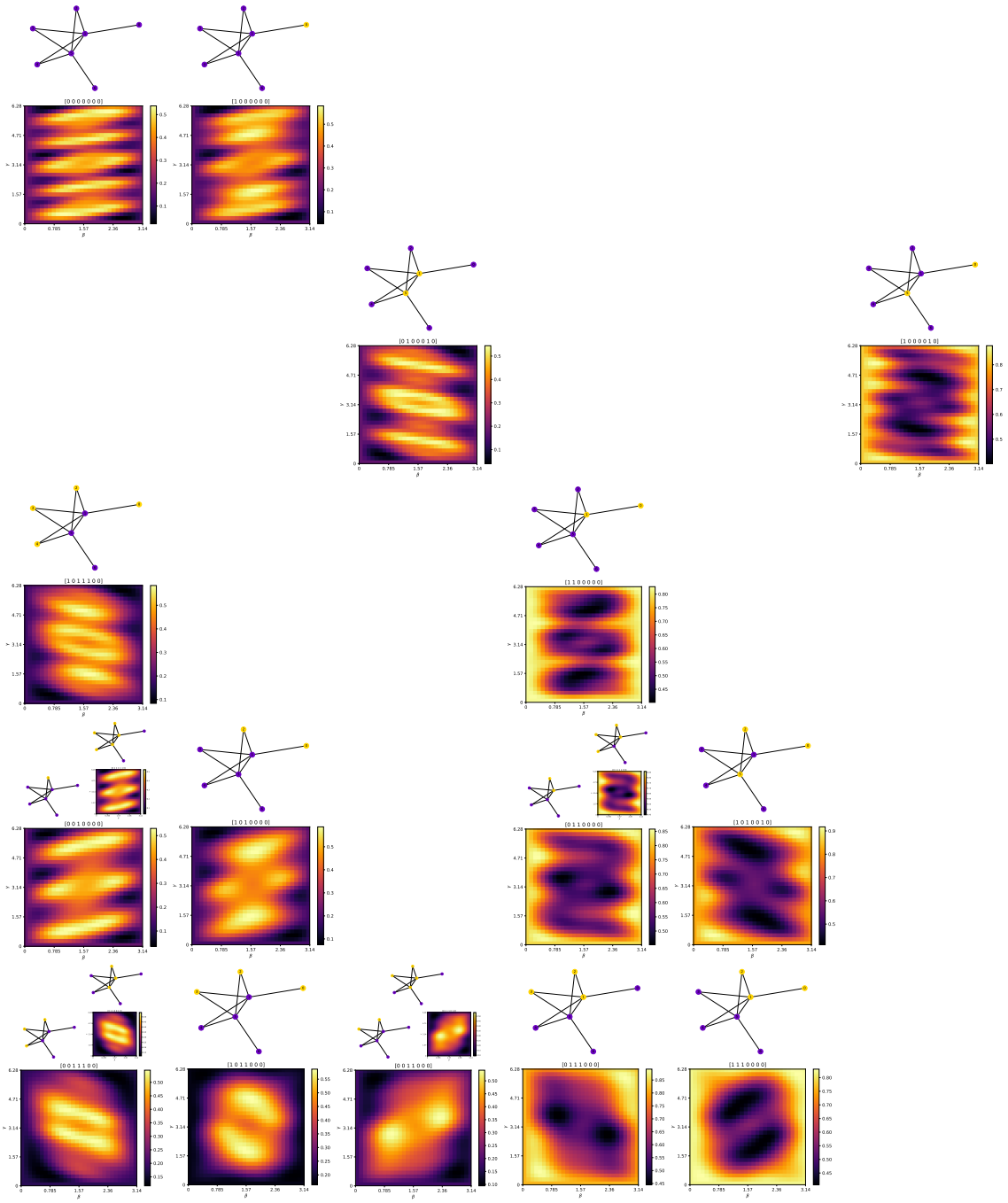

Figure A.8: All different initial cuts for graph 5reg3. The ordering is the same as in Figure A.5.

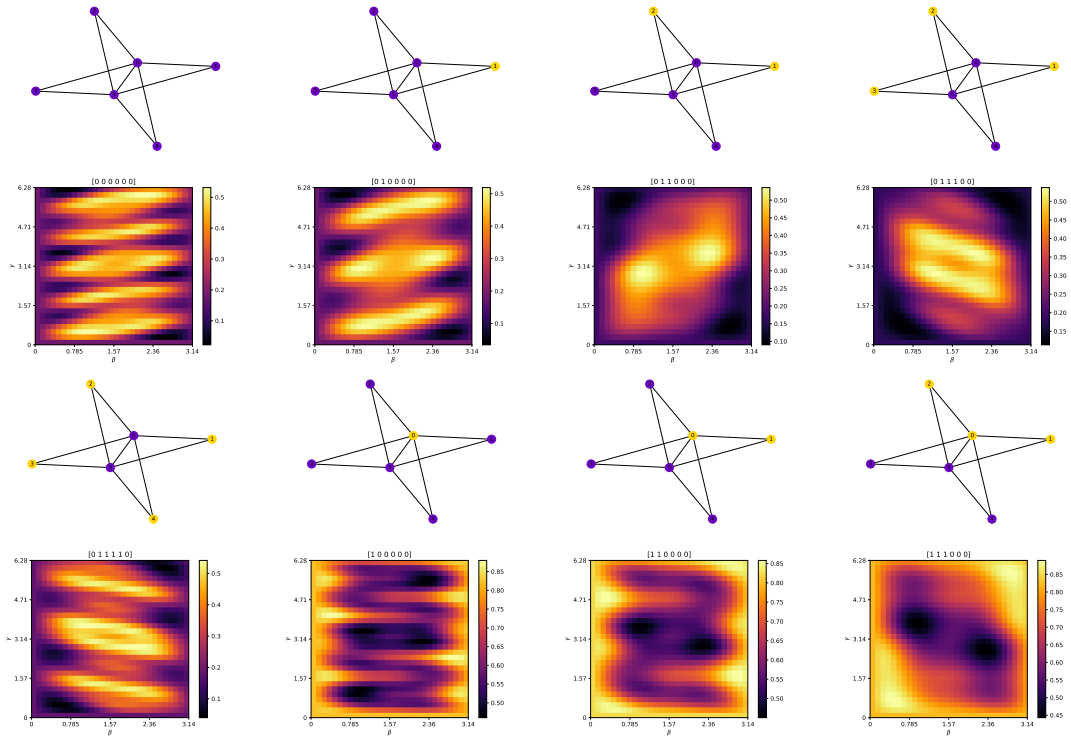

Figure A.9: All different initial cuts for graph 5reg4.

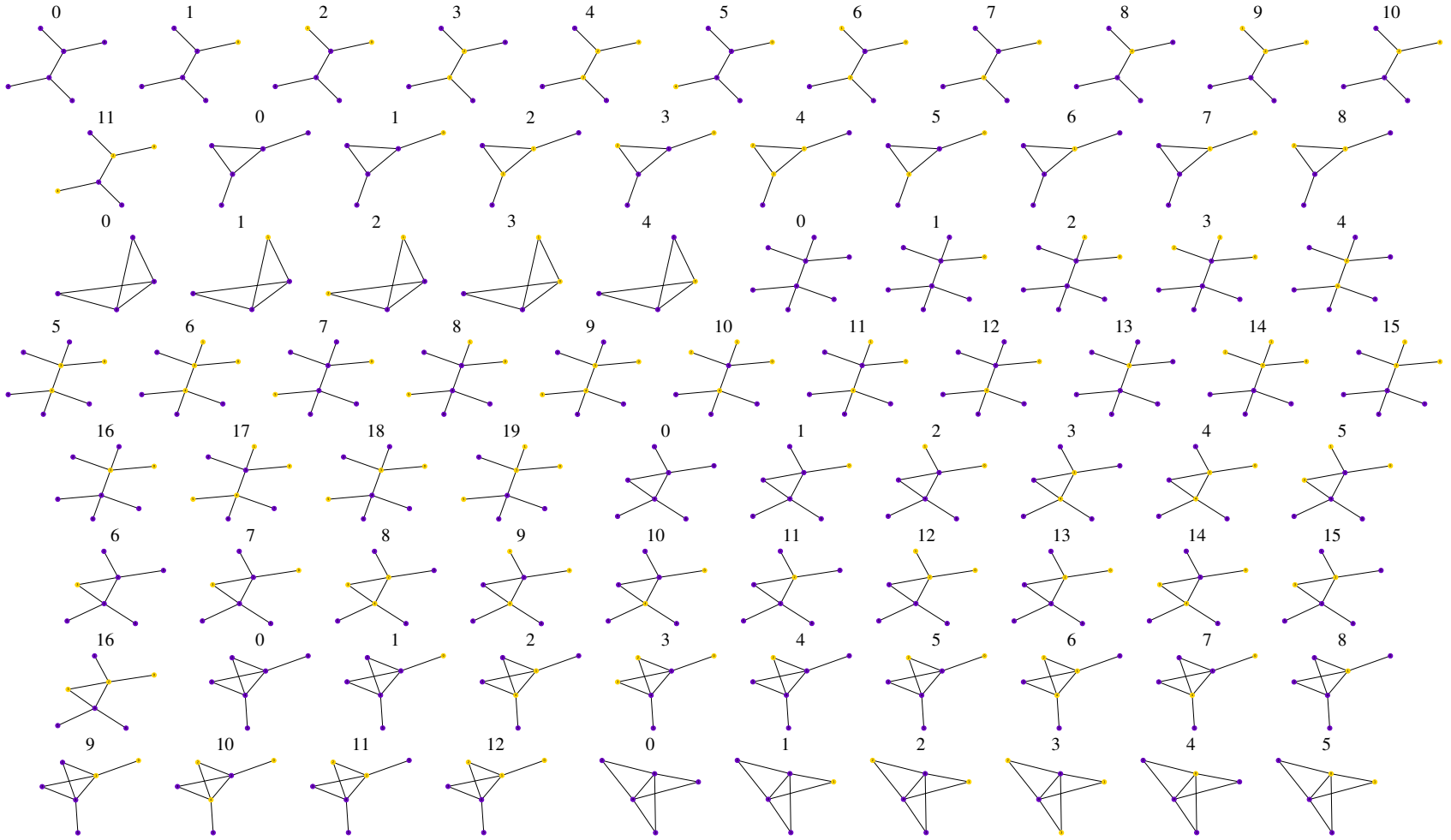

Figure A.10: All different initial cuts used for the regular graphs. The graphs are ordered in the way they appear in Figures 6.11 and 6.12. The numbering starts at 0 for each graph type. This figure serves as a reference to find the corresponding graph in a block of the aforementioned maps. (Shown here are the 3- and 4-subgraphs; 5-subgraphs on the following page.)

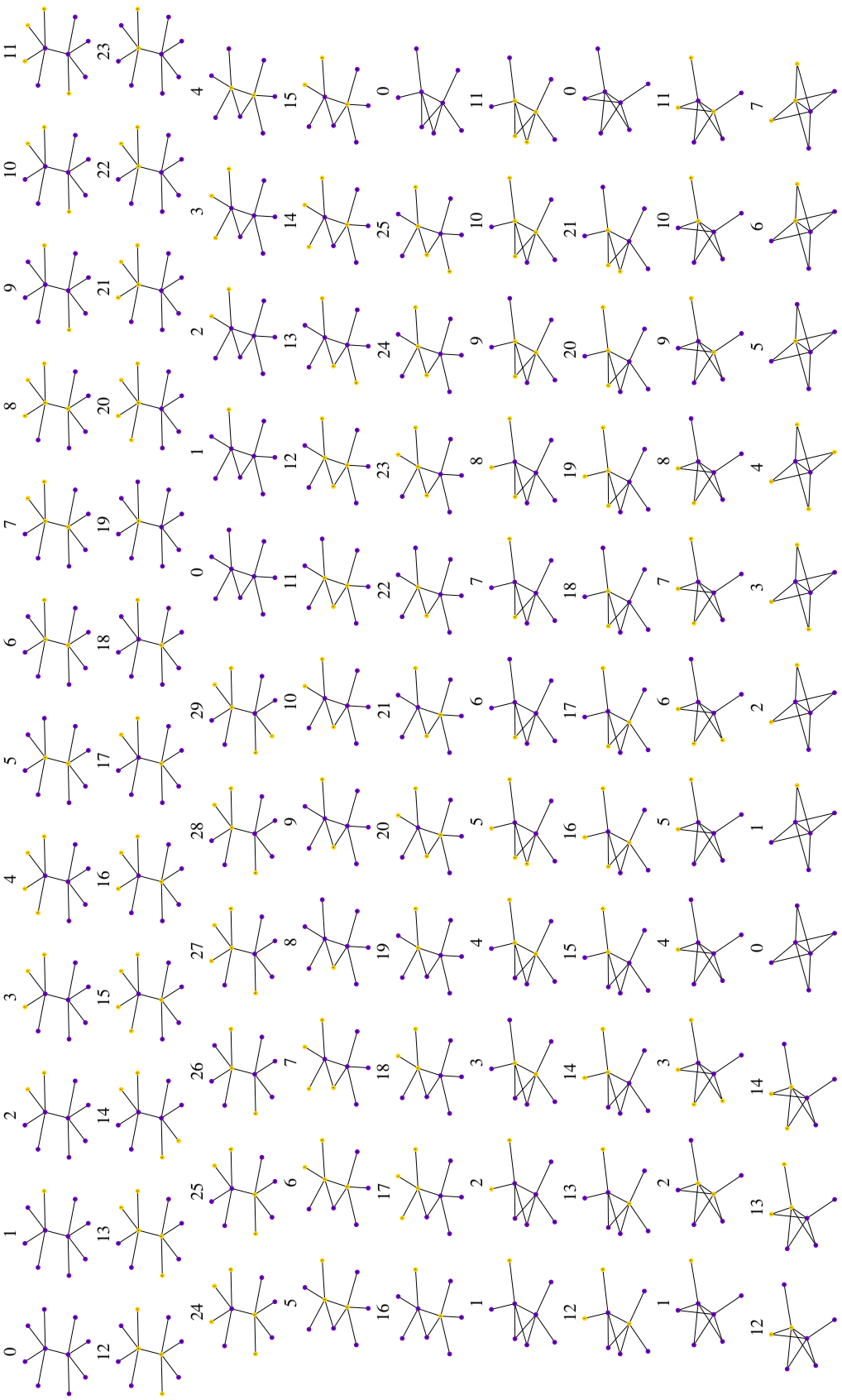

Figure A.10: All different initial cuts used for the regular graphs (cont.)

B Experiment Details

Parameter Concentration

Good donor parameters were obtained by optimization as described in Section 6.4. The optimization was performed on the subgraphs to save resources and computation time. As an optimizer COBYLA was chosen. To generate the graphs to investigate the parameter concentration the python library NetworkX [15] was used. The donors and acceptor instances were created by the following command:

```
G_donor = nx.random_regular_graph(x, 20, seed=y)
```

Where `x` is the number specifying the degree of every node and `y` is a seed used for reproducibility. The donors for every `x = 3, 4, 5` used `seed=0`. The other instances used the numbers 1 to 30, `seed=1...30` for each `x`. To calculate the energy of the acceptors, the circuit resulting from the whole graph was simulated using Qiskit's `'statevector_simulator'`.

NumPy's [16] capabilities to generate random numbers were the source of the random parameters:

```
np.random.seed(42)
gamma, beta = np.random.rand(2) * np.array([2*np.pi, np.pi])
```

A complete list of the random parameters can be found in the repository [19] as a `.csv` file.

The approximate cuts were generated with the GW algorithm. All the random cuts are displayed in Tables B.1 to B.3 The first row in each table is the seed used for generating the random graph. Note, that entry 0 is the donor. Next is a string representing the cut used for warm-starting. The first number is the initialization for node 0, the second for node 1 and so on. Next to every approximate cut is the value of this cut and the value of the true MaxCut followed by the performance of the parameters measured as the ratio of total energy to true MaxCut.

Landscape Approximation

The random 4-regular graph with 20 nodes was generated by using `seed=1234`. The approximate cut generated by the GW-algorithm was 11011101010010001000 and had a value of 32 with 34 being the true MaxCut value.

3-regular

seed	GW-cut	value	MaxCut	Low	Random	High
0	01011110110010010101	25	26	0.483	-	0.930
1	00101010011110001110	26	26	0.476	0.787	0.969
2	10011001001110010011	26	26	0.479	0.591	0.970
3	11100010100001111100	26	26	0.482	0.801	0.970
4	00101101000010111110	26	26	0.499	0.686	0.971
5	01010000001110111100	25	27	0.490	0.753	0.905
6	00010110110000011011	25	27	0.478	0.777	0.913
7	11001110100101101000	26	26	0.502	0.609	0.971
8	01111110000100101001	24	26	0.495	0.685	0.908
9	10001111100010110101	25	27	0.486	0.640	0.905
10	10010010101100110010	25	26	0.505	0.721	0.940
11	11000100110011010010	25	26	0.490	0.650	0.947
12	00100100011101001110	25	27	0.470	0.625	0.904
13	01011001110011100100	24	28	0.469	0.511	0.844
14	11010010111100011000	26	27	0.482	0.526	0.943
15	01101100100101100011	26	27	0.467	0.745	0.934
16	00100110111010010101	26	26	0.492	0.636	0.970
17	00011010111110010001	26	27	0.470	0.744	0.942
18	01010101111011000100	26	26	0.479	0.939	0.970
19	11000011010101100010	23	25	0.519	0.798	0.904
20	00000010111101111011	25	27	0.462	0.653	0.919
21	00110011010100111010	24	28	0.477	0.542	0.845
22	01011100001000111101	26	26	0.473	0.773	0.969
23	01001101011110110100	25	26	0.494	0.697	0.947
24	11011001010011000011	26	27	0.473	0.650	0.942
25	10100010101100111100	26	27	0.479	0.556	0.942
26	11010111110000110010	23	26	0.518	0.839	0.878
27	01111011101010000101	25	27	0.492	0.888	0.905
28	00110011101000111011	25	26	0.479	0.897	0.930
29	10111010010001100100	27	27	0.461	0.943	0.971
30	11010010100101100110	28	28	0.450	0.898	0.974

Table B.1: Experiment details on parameter concentration for 3-regular warm-started graphs. The donor graph is seed 0. The most significant bit of the GW-cut is node 0. Next to each cut is its value and the value of the true MaxCut. The three last columns show the approximation ratios, i.e. energy divided by MaxCut when transferring parameters of low, random and high objective function value.

4-regular

seed	GW-cut	value	MaxCut	Low	Random	High
0	10110111001010001010	32	32	0.551	-	0.969
1	11001011010110100001	28	32	0.572	0.810	0.856
2	11100011101000100010	34	34	0.506	0.655	0.972
3	00101001001001111110	28	34	0.511	0.665	0.823
4	01001111001100100000	32	32	0.528	0.684	0.978
5	01101001000101010111	32	32	0.549	0.563	0.964
6	00110001010011110011	36	36	0.487	0.820	0.970
7	11100101111100100010	28	30	0.581	0.703	0.943
8	01011010100101100011	30	32	0.546	0.692	0.919
9	00001111000101110011	30	32	0.554	0.619	0.916
10	00011001010110110011	32	34	0.536	0.593	0.913
11	10000001111101010110	32	34	0.527	0.824	0.920
12	10110111110010010100	30	34	0.531	0.606	0.860
13	01010011100011001001	32	32	0.546	0.960	0.971
14	10010011101010001011	30	34	0.528	0.605	0.869
15	00101010010111000111	30	32	0.557	0.867	0.911
16	10111000101101111010	30	32	0.530	0.843	0.926
17	00111011111000110000	34	34	0.519	0.778	0.970
18	01110101101110000100	34	34	0.517	0.957	0.972
19	00001101100011101110	30	32	0.558	0.804	0.914
20	00111011100000101101	30	32	0.539	0.592	0.915
21	01010110001011101000	34	34	0.514	0.529	0.975
22	10110100100010100110	32	32	0.540	0.787	0.971
23	00010111111101110000	30	32	0.549	0.675	0.915
24	00001111001101111010	32	34	0.519	0.577	0.917
25	11010101100101010100	32	34	0.512	0.914	0.917
26	11100010100110111100	32	32	0.549	0.945	0.972
27	01101001110100011111	30	32	0.541	0.918	0.929
28	00110111111010001101	30	34	0.494	0.684	0.865
29	11011100001010101001	34	34	0.509	0.905	0.969
30	11110100011011000001	30	32	0.563	0.853	0.914

Table B.2: Experiment details on parameter concentration for 4-regular warm-started graphs. The donor graph is seed 0. The most significant bit of the GW-cut is node 0. Next to each cut is its value and the value of the true MaxCut. The three last columns show the approximation ratios, i.e. energy divided by MaxCut when transferring parameters of low, random and high objective function value.

5-regular

seed	GW-cut	value	MaxCut	Low	Random	High
0	11010001110010101010	40	42	0.541	-	0.924
1	01111101010001010100	34	40	0.556	0.753	0.846
2	10110011011110100010	37	39	0.572	0.660	0.928
3	10100101010100111001	38	38	0.598	0.789	0.969
4	00000011001101001101	36	40	0.536	0.642	0.893
5	11100001010100110110	36	38	0.603	0.798	0.923
6	00001001110110011011	40	40	0.555	0.847	0.969
7	01101100000110111100	38	38	0.594	0.746	0.968
8	00101100110101001001	39	39	0.559	0.745	0.968
9	11001110000010100110	39	40	0.554	0.651	0.954
10	01010110101110010000	35	42	0.550	0.712	0.817
11	01001000111110001000	36	41	0.531	0.720	0.879
12	10110110010000111001	38	39	0.569	0.660	0.949
13	01010011110110001110	41	41	0.528	0.600	0.969
14	00100110111100011110	39	41	0.543	0.638	0.927
15	01001101111100101000	38	39	0.580	0.809	0.949
16	00100100110101101010	39	39	0.565	0.757	0.968
17	10110000110010110101	38	39	0.585	0.793	0.944
18	01100101010111010100	40	40	0.554	0.957	0.969
19	01101010000101110101	38	39	0.577	0.813	0.949
20	01010001011110000101	37	39	0.569	0.654	0.924
21	00110111001101011010	37	39	0.572	0.602	0.923
22	11110001100100101100	40	40	0.548	0.777	0.969
23	11010000101110110110	39	41	0.542	0.651	0.931
24	11001111000000110011	36	38	0.599	0.690	0.927
25	10011001100010111011	37	40	0.553	0.615	0.910
26	11000100101111010001	38	40	0.568	0.907	0.925
27	01010011001011001001	35	39	0.584	0.872	0.879
28	10010110110001001010	41	42	0.528	0.890	0.952
29	11100001101011101000	40	40	0.552	0.876	0.969
30	01111001100010101000	37	40	0.560	0.835	0.905

Table B.3: Experiment details on parameter concentration for 5-regular warm-started graphs. The donor graph is seed 0. The most significant bit of the GW-cut is node 0. Next to each cut is its value and the value of the true MaxCut. The three last columns show the approximation ratios, i.e. energy divided by MaxCut when transferring parameters of low, random and high objective function value.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature