Bachelorarbeit

Lösen von Optimierungsproblemen aus dem Supply Chain Management mit Quantenalgorithmen

Fachbereich Elektrotechnik und Informationstechnik
FH Aachen

vorgelegt von
Paul Benedikt Franz Hüppe
geboren am 28. November 1995 in Duisburg

Erstprüfer: Prof. Dr. rer. nat. Dr.-Ing. Georg Hoever
Zweitprüfer: Dr. Dennis Willsch

Aachen, 17. Oktober 2025

Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe.

Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht.

Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen.

Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Aachen, im Oktober 2025

Zusammenfassung

Quantencomputer bieten die Möglichkeit, einige Probleme effizienter zu lösen als klassische Computer. Im Supply Chain Management (SCM) und in der Logistik gibt es viele Probleme, die hohen Rechenaufwand erfordern. Diese Arbeit soll die Umsetzbarkeit von Quantenalgorithmen in diesen Bereichen untersuchen und deren Potential bewerten.

Der Fokus liegt auf Optimierungsproblemen, die als Quadratic Unconstrained Binary Optimization Probleme formuliert werden können, wie z.B. Traveling Salesman Probleme oder Rucksack-Probleme. Insbesondere sollen Optimierungsprobleme betrachtet werden, die in dem Bereich der Bedarfsprognose im Supply Chain Management auftreten.

Die Lösung dieser Probleme soll mit klassischen Algorithmen auf herkömmlichen Computern und auch auf einem Quantenannealer getestet und bewertet werden. Ziel ist es, die Umsetzbarkeit in SCM-Prozessen festzustellen.

Zur Validierung werden Problemstellungen aus dem Alltag im Supply Chain Management von ALDI SÜD Dienstleistungs-SE & Co. oHG betrachtet, um einen praktischen Bezug herzustellen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

1 Einleitung und Aufgabenstellung

Es gab in den letzten Jahren erhebliche Fortschritte auf dem Gebiet der Quantencomputertechnologie. Diese neue Art des Rechnens verspricht, bestimmte Probleme schneller und effizienter lösen zu können als herkömmliche Computer [13]. Im Tagesgeschäft des Supply Chain Management (SCM) und der Logistik treten zahlreiche Optimierungsprobleme auf, die durch den Einsatz von Quanten Computing (QC) profitieren könnten. Besonders hier sind schnelle Lösungen wichtig, um einen effizienten und profitablen Ablauf der Lieferketten und des Warenflusses zu ermöglichen. In dieser Arbeit soll untersucht werden, wie Quantenalgorithmen zur Lösung derartiger Probleme beitragen können und welche potentiellen Auswirkungen auf zukünftige Business-Prozesse daraus entstehen können.

2 Grundlagen

2.1 Binäre Optimierungsprobleme

Binäre Optimierungsprobleme können auf einem Quantenannealer (QA) oder mithilfe des Quantum Approximate Optimization Algorithm (QAOA) [9] auf Gatter-Basierter Quantencomputer (GBQC) gelöst werden.

Es gibt zwei äquivalente mathematische Formulierungen von binären Optimierungsproblemen:

Zum Einen kann das Problem als Ising-Hamiltonian formuliert werden. Die zu optimierende Kosten- / Energiefunktion hat hier allgemein die Form:

$$H(\mathbf{s}) = \sum_{i=0}^{N-1} h_i s_i + \sum_{i<j}^{N-1} J_{ij} s_i s_j, \quad s_i \in \{-1, 1\} \tag{2.1}$$

Zum Anderen kann das Problem als Quadratic Unconstrained Binary Optimization Problem (QUBO) in Form einer Matrix Q angegeben werden:

$$H(\mathbf{x}) = \left(\sum_{i=0}^{N-1} a_i x_i + \sum_{i<j}^{N-1} b_{ij} x_i x_j \right) = \left(\sum_{i \leq j}^{N-1} x_i Q_{ij} x_j \right), \quad x_i \in \{0, 1\} \tag{2.2}$$

Zwischen den beiden Formulierungen können die Problemvariablen mittels $x_i = (1 + s_i)/2$ leicht transformiert werden.

In der QUBO Matrix Q_{ij} werden die Kosten und Nebenbedingungen des Problems codiert und erfasst. Intuitiv bedeutet dies, dass der Kostenbeitrag Q_{ij} entsteht, wenn $x_i = 1$ und $x_j = 1$ als Teil der Lösung gefunden werden.

Lösung des Optimierungsproblems ist also der Vektor $\mathbf{x}$, für den die Kostenfunktion minimal ist (2.3).

$$\underset{\mathbf{x}}{\operatorname{argmin}} \sum_{i \leq j} x_i Q_{ij} x_j \tag{2.3}$$

Für die in dieser Arbeit modellierten Optimierungsprobleme aus dem SCM wird die Formulierung als QUBO-Matrix mit binären Variablen x_i verwendet.

2.2 Nebenbedingungen und Strafterme

Letztendlich setzt sich der Gesamt-Hamiltonian H_P des QUBOs aus der initialen Kostenfunktion, sowie den Nebenbedingungen des Problems zusammen. Um die Einhaltung der Nebenbedingungen zu gewährleisten, wird ein entsprechender Strafterm mit einem Skalierungsfaktor λ (Lagrange-Parameter) zur Kostenfunktion hinzuaddiert. Dadurch wird sichergestellt, dass ein Nichteinhalten der Nebenbedingung zu einer Erhöhung der Kosten führt. Wenn ein Lösungs-Zustand die Nebenbedingung erfüllt, so ist auch der Strafterm hierfür minimal. Eine Verschiebung um die sogenannte Offset-Energie kann dieses Minimum zu 0 gesetzt werden. Das erleichtert das schnelle Identifizieren von gültigen Lösungen.

$$H_\mathrm{P}(\mathbf{x}) = H_\mathrm{Kosten}(\mathbf{x}) + \sum_i \lambda_i H_{\mathrm{NB}_i}(\mathbf{x}) \tag{2.4}$$

Die genaue Form des Strafterms hängt von der Art der Nebenbedingung ab. In Tabelle 2.1 sind einige grundlegende Nebenbedingungen mit binären Variablen und mögliche Strafterme aufgeführt.

Tabelle 2.1: Eine Auswahl an Nebenbedingungen und dazugehörige bekannte Strafterme nach [11].

Klassische Nebenbedingung	Möglicher QUBO-Strafterm
$x_i + x_j \leq 1$	$\lambda(x_i x_j)$
$x_i + x_j \geq 1$	$\lambda(1 - x_i - x_j + x_i x_j)$
$x_i + x_j = 1$	$\lambda(1 - x_i - x_j + 2x_i x_j)$
$x_i \leq x_j$	$\lambda(x_i - x_i x_j)$
$x_i + x_j + x_k \leq 1$	$\lambda(x_i x_j + x_i x_k + x_j x_k)$
$x_i = x_j$	$\lambda(x_i + x_j - 2x_i x_j)$

Andere Strafterme sind auch möglich, solange diese genau dann minimal sind, wenn die Nebenbedingung erfüllt ist, und sonst einen positiven Beitrag zu den Gesamtkosten geben.

Auch komplexere Nebenbedingungen, wie z.B. die Ungleichung (2.5), können unter Zuhilfenahme sogenannter Slack-Variablen (und somit zusätzlichen Qubits) als quadratische Terme realisiert werden.

$$\sum_{i=1}^{n} l_i x_i \leq B, \quad l_i \in \mathcal{Z} \tag{2.5}$$

Die in dieser Arbeit betrachteten Probleme benötigen keine Ungleichungen als Nebenbedingung, daher werden diese auch nicht weiter erläutert.

2.3 Codierung der Problemvariablen

Nicht jedes Optimierungsproblem, welches mittels Quanten Annealing (QA) gelöst werden soll, hat binäre Problemvariablen. Demnach müssen reelle/ganze Zahlen codiert werden, um sie für eine QUBO-Formulierung nutzen zu können. Jede Wahl einer Codierung erfordert zusätzliche Terme in den Nebenbedingungen, die erzwingen, dass die Codierung der Problemvariablen eingehalten wird. Mögliche Optionen für die Codierung einer ganzzahligen Variable $v \in \mathbb{N}$ sind zum Beispiel:

- Integer Encoding: Ganzzahlige Werte der Problemvariable v werden durch ihre binäre Integer Darstellung codiert.

  Beispiel:

  $$v = \sum_{i=0}^{N-1} 2^i x_i \tag{2.6}$$

  Diese Codierung ist zwar die effizienteste bezogen auf die Anzahl der Qubits, bedingt im Allgemeinen aber auch sehr komplexe Terme um die Wechselwirkung zweier Variablen beschreiben zu können. [6]

- One-Hot Encoding: Jeder mögliche Wert, den die Problemvariable v annehmen kann, wird durch ein Bit dargestellt. Hat dieses Bit den Wert 1 so wird der Index als Wert der Integer Variablen v zugewiesen.

  $$x_i = \begin{cases} 1, & i = v \\ 0, & \text{sonst} \end{cases}, \quad \text{bzw.} \quad v = \sum_{i=0}^{N-1} i x_i \tag{2.7}$$

  Die Anzahl der möglichen Werte, die $v \in \{0, \ldots, N-1\}$ annehmen kann, gibt also auch die Anzahl der benötigten Qubits an ($\#qubits = N$).

  Es muss zudem auch sichergestellt sein, dass immer nur genau ein Bit aus dem Bitstring 1 ist. Andernfalls besteht keine eindeutige, gültige Codierung. Daraus ergibt sich die Nebenbedingung (2.8) und der dazugehörige Strafterm (2.9).

  $$\sum_i x_i = 1 \tag{2.8}$$

  $$H_{\text{One-Hot}} = \left(1 - \sum_i x_i\right)^2 \tag{2.9}$$

  Beispiel: $v \in \{0, 1, 2, 3\}$ wird codiert als x_0, x_1, x_2, x_3 mit Nebenbedingung $x_0 + x_1 + x_2 + x_3 = 1$.

  $$v = 2 \rightarrow x_2 = 1, \quad x_0 = x_1 = x_3 = 0 \tag{2.10}$$

  $$v = \sum_{i=0}^{N-1} i x_i \tag{2.11}$$

  Die One-Hot Codierung benötigt die meisten Qubits der vorgestellten Methoden und setzt zudem eine All-To-All Konnektivität der Qubits dieses Teil-Bitstrings voraus,

um die Nebenbedingung dieser Codierung sicherzustellen. Sie ist jedoch sehr intuitiv auf eine Vielzahl von Problemen anwendbar.

- Domain-Wall Encoding [6]: Die Position eines Bit-Flips innerhalb eines Bitstrings codiert den Wert der Problemvariablen v.

  Beispiel: $v \in \{0, 1, 2, 3\}$ wird codiert in den Qubits x_0, x_1, x_2 mit

$$v = 2 \qquad \rightarrow \qquad x_i = \begin{cases} 1, & i < 2 \\ 0, & sonst \end{cases} \tag{2.12}$$

$$v = \sum_{i=0}^{N-1} i(x_{i-1} - x_i) \tag{2.13}$$

  Dabei sind die gedachten Rand-Bits fixiert und als $x_{-1} = 1$ und $x_{N-1} = 0$ definiert. Die Nebenbedingung zur Einhaltung der Domain-Wall Codierung lautet

$$H_{\mathrm{DW}} = \sum_{i=-1}^{N-2} 1 - 2x_i - 2x_{i+1} + 4x_i x_{i+1} \tag{2.14}$$

  Vorteile sind zum Einen, dass im Vergleich zur One-Hot Codierung pro Variable v ein Qubit weniger benötigt wird, und zum Anderen, dass die Nebenbedingung nur paarweise Konnektivität zwischen benachbarten Qubits (i und $i + 1$) in dem Bitstring erfordert.

2.4 Separierung von Termen höherer Ordnung

Nicht jedes Problem lässt sich direkt in einer Formulierung mit nur quadratischen Termen beschreiben. Terme höherer Ordnung in den Problemvariablen (z.B. der Form $x_i x_j x_k$) sind erst einmal nicht mit dem QUBO-Formalismus vereinbar. Durch die Einführung von zusätzlichen Hilfsqubits und weiterer Nebenbedingungen können diese Terme in eine kompatible quadratische Form gebracht werden.

Hierzu wird ein weiteres Qubit als $q = x_i x_j$ definiert. Dieses hat genau dann den Wert 1, wenn das Produkt $x_i x_j$ auch den Wert 1 hat. Um die Bindung des Hilfsqubits an die Problemvariablen zu gewährleisten wird ein zusätzlicher Strafterm (2.15) eingefügt. Das Produkt $x_i x_j$ in der Kostenfunktion wird durch das Hilfsqubit q substituiert (s. Gleichung (2.16)).

$$H_{\text{Sep}} = x_i x_j - 2x_i q - 2x_j q + 3q^2 \tag{2.15}$$

$$x_i \cdot x_j \cdot x_k = q \cdot x_k \tag{2.16}$$

In Tabelle 2.2 ist leicht zu erkennen, dass dieser Strafterm genau dann minimal (sogar 0) ist, wenn die Substitution $q = x_i x_j$ erfüllt ist.

Tabelle 2.2: Wahrheitstabelle der Nebenbedingung für Hilfsqubit q.

x_i	x_j	q	NB
0	0	0	0
0	0	1	3
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

2.5 Quanten Annealing

Der D-Wave Quantum Annealer transformiert und nutzt für die Berechnung das Ising Modell, da dieses sich direkt mithilfe von elektromagnetischen Feldern auf den Quantenbits umsetzen lässt. Die elektromagnetischen Felder sind durch die Koeffizienten h_i und J_{ij} aus Gleichung (2.1) bestimmt. Die h_i Terme wirken hierbei direkt auf nur ein einzelnes Qubit i, wohingegen die Terme J_{ij} die Kopplung bzw. Wechselwirkung zwischen den Qubits i und j beschreiben. Ausgehend von einem bekannten Energie-Grundzustand H_{init} wird das System „langsam“ zu dem Energie-Zustand H_P des Problems überführt (2.17), wobei s die skalierte Zeit t/T_{anneal} ist. Im Falle des D-Wave `Advantage_system5.4` ist dieser initiale Hamiltonian gegeben durch Gleichung (2.18).

$$H(s) = A(s)H_{\text{init}} + B(s)H_{\text{P}} \tag{2.17}$$

$$H_{\text{init}} = -\sum_{i=1}^{N} \sigma_i^x \tag{2.18}$$

Die Parameter $A(s)$ und $B(s)$ geben den Annealing Schedule an. Dieser gibt an, wie schnell und zu welchen Anteilen von dem initialen Hamiltonian H_{init} zu dem Problem Hamiltonian H_{P} überblendet wird. Der Standard Annealing Schedule aus dem Datenblatt des D-Wave `Advantage_system5.4` [20] ist in Abbildung 2.1 gezeigt. Geschieht der Annealing-Prozess langsam genug, verbleiben die Qbits idealerweise im Energie Grundzustand des gerade wirkenden Hamiltonian gemäß der adiabatischen Theorie. [10]

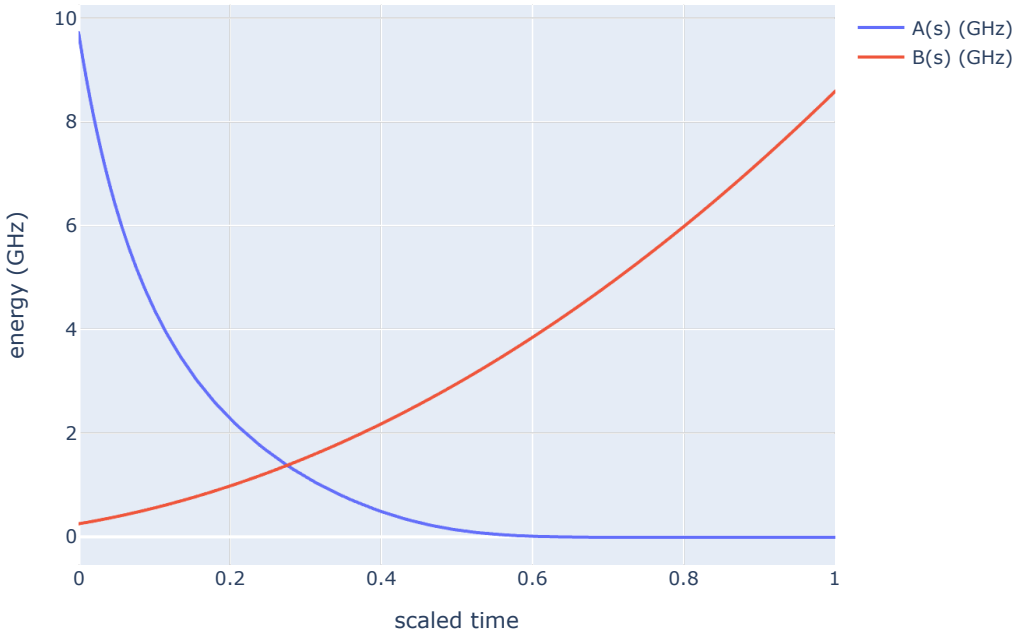

Abbildung 2.1: Der Standard Annealing Schedule des D-Wave `Advantage_system5.4`.

Der Annealing Vorgang auf dem D-Wave `Advantage_system5.4` kann durch Gleichung (2.19) beschrieben werden.

$$H(s) = -\frac{A(s)}{2}\left(\sum_i \hat{\sigma}_x^{(i)}\right) + \frac{B(s)}{2}\left(\sum_i h_i \hat{\sigma}_z^{(i)} + \sum_{i>j} J_{i,j}\hat{\sigma}_z^{(i)}\hat{\sigma}_z^{(j)}\right) \tag{2.19}$$

Nach Abschluss des Annealing-Vorgangs ($s = 1$, $A(1) \ll B(1)$) werden die Qbits, auf welche die Pauli-Operatoren $\hat{\sigma}_z^{(i,j)}$ nun wirken, gemessen. Daraus ergeben sich die Problemvariablen $s_i \in \{-1, 1\}$ in Gleichung (2.1) des Ising-Problems. Sie repräsentieren nun theoretisch einen Zustand der minimalen Energie unter Einfluss des Problem-Hamiltonians.

Aktuell befinden wir uns in der NISQ (Noisy Intermediate-scale Quantum) Ära, weswegen das Finden der minimalen Energie nicht garantiert ist. Durch mehrfaches Annealen des selben Hamiltonians kann aber eine Verteilung von niederenergetischen Zuständen erlangt werden.

2.6 (Bedingte) gemeinsame Information

Die gemeinsame Information (auch gegenseitige oder Trans- Information) zwischen zwei Kennzahlen/Zufallsgrößen ist ein Maß für die Stärke des statistischen Zusammenhangs dieser Kennzahlen/Zufallsgrößen. Für diskrete Zufallsgrößen (X, Y) kann sie aus den diskreten Wahrscheinlichkeitsverteilungen p_i wie in Gleichung (2.20) definiert werden [7]:

$$I(X;Y) = \sum_{y \in \mathcal{Y}} \sum_{x \in \mathcal{X}} P_{(X,Y)}(x,y) \log \left(\frac{P_{(X,Y)}(x,y)}{P_X(x)\, P_Y(y)} \right) \tag{2.20}$$

Die bedingte gemeinsame Information ist das Maß des statistischen Zusammenhangs, wenn bereits eine andere Kennzahl/Zufallsgröße mit ggf. unbekanntem Zusammenhang zu den anderen Kennzahlen/Zufallsgrößen bekannt ist. Analog zur gemeinsamen Information wird sie im diskreten Fall definiert (s. Gleichung (2.21)) aus den Wahrscheinlichkeitsverteilungen [7].

$$I(X;Y|Z) = \sum_{z \in \mathcal{Z}} \sum_{y \in \mathcal{Y}} \sum_{x \in \mathcal{X}} P_{(X,Y,Z)}(x,y,z) \log \left(\frac{P_Z(z)\, P_{(X,Y,Z)}(x,y,z)}{P_{(X,Z)}(x,z)\, P_{(Y,Z)}(y,z)} \right) \tag{2.21}$$

Es gibt mehrere Möglichkeiten die gemeinsame Information zweier Kennzahlen zu berechnen. Hierbei ist es wichtig, ob es sich bei den Datenreihen um diskrete, kategorische oder kontinuierliche Messgrößen handelt. Nicht jede Methode ist für jede Kombination geeignet.

- Diskretisierung von nicht kontinuierlichen Datenreihen und anschließende Bestimmung der diskreten Wahrscheinlichkeitsverteilung anhand von Histogrammen mit einer endlichen Anzahl von Bins. Hierbei werden die Definitionen (2.20) und (2.21) angewendet.
- Approximation über k nächste Nachbarn [15] in den Datenreihen. Dies kann als adaptive Partitionierung aufgefasst werden.
- Berechnung oder Approximation der Entropie $H(X)$ und Verbundentropie $H(X|Y)$ der Datenreihen.
  Anschließend kann die gemeinsame Information zwischen Feature X und einem Target Y durch $I(X;Y) = H(X) - H(X|Y)$ nach [7] berechnet werden.
  Die bedingte gemeinsame Information zwischen Feature X und Target Y mit bekanntem Feature Z ergibt sich dann aus [7] zu $I(X;Y|Z) = H(X|Z) - H(X|Y,Z)$.

  Für die Bestimmung der Entropie H können wiederum verschiedenste Methoden verwendet werden, auf die hier nicht weiter eingegangen wird.

3 Anwendungsfall: Modellierung einer Mischpalette

Auf einer Lage Joghurt im Supermarkt werden häufig mehrere Sorten kombiniert (z.B. Vanille, Kirsche und Erdbeere). Diese Mischpalette wird in allen Filialen in der gleichen Zusammensetzung zum Verkauf angeboten. Das Kaufverhalten der Kunden kann mitunter starke regionale Unterschiede aufweisen. Für die Bedarfsplanung ist es eine essenzielle Frage, zu welchen Anteilen welche Sorte auf einer Lage vorhanden sein soll. Ungünstig wäre es, wenn ein Kunde sein gewünschtes Lieblingsprodukt nicht erwerben kann, aber ebenso unvorteilhaft wäre das Erreichen des Mindesthaltbarkeitsdatum von nicht verkauften Sorten. Daher stellt sich die Frage, ob eine optimale Zusammenstellung der Sorten gefunden werden kann, bei der gleichzeitig der Abverkauf optimiert als auch die Abschriften von unverkaufter Ware minimiert werden können.

Man betrachte $\mathcal{F}$ als eine Menge von F Filialen, die modelliert werden sollen, $\mathcal{S}$ sei die Menge an S verschiedenen Sorten Joghurt, mit der die Palette bestückt werden kann. Betrachtet wird ein fiktiver Zeitraum, der so gewählt ist, dass eine neue Lieferung genau dann eintrifft, wenn die alte Lieferung ohne Beachtung der Sorten komplett abverkauft wäre.

3.1 Codierung der Problemvariablen

Die Problemvariablen sind in diesem Fall die Zusammensetzung der Mischpalette und die Liefermenge in ganzen Paletten an eine Filiale. Entweder kann die absolute Anzahl an Produkten einer Sorte auf einer Palette, oder ein Anteil an der Gesamtmenge als Problemvariable verwendet werden. Im Folgenden wird die absolute Menge an Joghurts dieser Sorte betrachtet und durch die Variablen $x_s,\ s \in \mathcal{S}$ dargestellt. Weiterhin ist der vorhandene Bestand in der Filiale abhängig von der Liefermenge an Paletten an diese Filiale. Diese Liefermenge wird mit $p_f,\ f \in \mathcal{F}$ dargestellt.

Beide ganzzahligen Variablen werden im Folgenden mit One-Hot (vgl. Seite 4) codiert. Für jedes p_f werden $I+1$ Qubits genutzt. I ist die maximale Anzahl an Paletten, die an eine einzelne Filiale geliefert werden kann. Für jedes x_s werden $J+1$ Qubits genutzt, wobei J die Anzahl an Joghurtbechern auf einer Palette angibt. Minimal sind 0 Plätze der Palette von dieser Sorte belegt, maximal sind es J und die Palette ist nur mit dieser Sorte gefüllt.

Es ergeben sich folgende Nebenbedingungen für die Codierung der Variablen:

$$H_{\text{One-Hot}-p} = \sum_{f\in\mathcal{F}} \left(1 - \sum_{i=0}^{I} (p_f)_i\right)^2 \tag{3.1}$$

$$H_{\text{One-Hot}-x} = \sum_{s\in\mathcal{S}} \left(1 - \sum_{j=0}^{J} (x_s)_j\right)^2 \tag{3.2}$$

3.2 Kostenfunktion für die Modellierung des Abverkaufs

Die Qualität einer Lösung wird mit einer Kostenfunktion bewertet, welche als Eingabe einen Bitstring der codierten Problemvariablen nimmt und daraus einen Wert für die Kosten dieser Lösung berechnet. Am globalen Optimum liefert sie den kleinsten Wert und umso größer die Abweichung von diesem ist, desto höher werden die Kosten.

Zur Modellierung des Abverkaufs wird der in einer Filiale f vorhandene Bestand jeder Sorte s errechnet aus:

$$\sum_{f\in\mathcal{F},s\in\mathcal{S}} p_f \cdot x_s. \tag{3.3}$$

Das Kundenverhalten wird durch die Absatzmatrix $N_{f,s}$ gegeben. Im Idealfall decken sich gelieferte Menge und Nachfrage genau, sodass in jeder Filiale für jede Sorte gilt: $N_{f,s} - p_f \cdot x_s = 0$.

Abweichungen von diesem Optimalfall sind unerwünscht und sollen demnach mit höheren Kosten versehen werden. Dabei soll der Fall, dass die Nachfrage nicht gedeckt werden kann ($N_{f,s} > p_f x_s$), gleichwertig zu dem Fall, dass Produkte unverkauft liegen bleiben ($N_{f,s} < p_f x_s$), mit einer Strafe belegt werden. Das geschieht allgemein durch die Berechnung (3.4).

$$H_{\text{Kosten}} = \sum_{f\in\mathcal{F},s\in\mathcal{S}} (N_{f,s} - p_f \cdot x_s)^2 \tag{3.4}$$

Setzt man die One-Hot Codierung der Variablen ein, ergibt sich (3.5)

$$H_{\text{Kosten}} = \sum_{f\in\mathcal{F},s\in\mathcal{S}} \left(N_{f,s}^2 - 2N_{f,s} \sum_{i=0}^{I} i(p_f)_i \sum_{j=0}^{J} j(x_s)_j + \left(\sum_{i=0}^{I} i(p_f)_i \sum_{j=0}^{J} j(x_s)_j \right)^2 \right) \tag{3.5}$$

3.3 Nebenbedingungen

Hinzu kommt noch eine Nebenbedingung, dass die Palette komplett gefüllt sein soll. Sind z.B. sechs Plätze vorhanden, so soll weder einer leer sein, noch sieben Produkte auf dieser platziert werden. Das heißt, dass die Summe der Sortenanteile der Größe der Palette entsprechen soll.

$$\sum_{s\in\mathcal{S}} x_s = J \tag{3.6}$$

$$H_{\text{NB}} = \left(\sum_{s\in\mathcal{S}} x_s - J \right)^2 = \left(\sum_{s\in\mathcal{S}} \left(\sum_{j=0}^{J} j(x_s)_j \right) - J \right)^2 \tag{3.7}$$

3.4 Separierung von Termen höherer Ordnung

Es ist offensichtlich, dass Terme der Form $(px)^2$ nicht mehr quadratisch sind in den Problemvariablen. Das ist erst einmal nicht mit dem QUBO Formalismus vereinbar, da hier die Terme maximal quadratischer Ordnung sein dürfen.

Um dennoch eine Formulierung als QUBO zu ermöglichen, kann die in Abschnitt 2.4 beschriebene Substitution angewendet werden: Für jeden Term höherer Ordnung in p und x wird ein weiteres Hilfsqubit eingeführt (3.8).

$$(q_{f,s})_{i,j} = (p_f)_i (x_s)_j \tag{3.8}$$

Dieses Hilfsqubit ist in dem Fall genau 1, wenn $p_f = 1$ und $x_s = 1$, da es sich um die binären Komponenten der codierten Problemvariablen handelt.

Um diese Bindung des Hilfsqubits an die Problemvariablen zu gewährleisten, wird eine zusätzliche Nebenbedingung eingefügt.

$$H_q = \sum_{f\in\mathcal{F}, s\in\mathcal{S}} \sum_{i,j} (p_f)_i \cdot (x_s)_j - 2(p_f)_i \cdot (q_{f,s})_{i,j} - 2(x_s)_j \cdot (q_{f,s})_{i,j} + 3(q_{f,s})_{i,j} \tag{3.9}$$

Das Einführen eines neuen flachen Index über die Indizes der Codierung i, j mit $k = i \cdot I + j$ verbessert die Übersichtlichkeit. Weiterhin gilt dann:

$$k \in \{0, \ldots, I \cdot J = K\} \tag{3.10}$$
$$i = \lfloor \tfrac{k}{I} \rfloor \tag{3.11}$$
$$j = k \bmod I \tag{3.12}$$
$$i \cdot j = \lfloor \tfrac{k}{I} \rfloor \cdot (k \bmod I) = \alpha(k) \tag{3.13}$$
$$(q_{f,s})_{i,j} = (q_{f,s})_k \tag{3.14}$$

Der gesamte Problem-Hamiltonian kann nun als (3.15) angegeben werden.

$$H_\mathrm{P} = H_\mathrm{Kosten} + \lambda H_\mathrm{NB} + \lambda_q H_\mathrm{q} + \lambda_x H_{\text{One-Hot}-x} + \lambda_p H_{\text{One-Hot}-p} \tag{3.15}$$

$$\begin{aligned}
&H_P(x,p,q) = \\
&\sum_{\substack{f\in\mathcal{F}\\ s\in\mathcal{S}}} \left[N_{f,s}^2 + \sum_{k=0}^{K} \left[\alpha(k)^2 - 2N_{f,s}\alpha(k)\right](q_{f,s})_k + 2\sum_{\substack{l=0\\ k<l}}^{K} \left[\alpha(k)\alpha(l)\right](q_{f,s})_k (q_{f,s})_l \right] \\
&\quad + \lambda \left(\sum_{s\in\mathcal{S}} \left(\sum_{j=0}^{J} j(x_s)_j \right) - J \right)^2 \\
&\quad + \lambda_q \sum_{\substack{f\in\mathcal{F}\\ s\in\mathcal{S}}} \sum_{i,j=0}^{I,J} (p_f)_i (x_s)_j - 2(p_f)_i (q_{f,s})_{i,j} - 2(x_s)_j (q_{f,s})_{i,j} + 3(q_{f,s})_{i,j} \\
&\quad + \lambda_x \sum_{s\in\mathcal{S}} \left(\sum_{j=0}^{J} (x_s)_j - 1 \right)^2 \\
&\quad + \lambda_p \sum_{f\in\mathcal{F}} \left(\sum_{i=0}^{I} (p_f)_i - 1 \right)^2
\end{aligned}$$

Die Lagrange-Parameter λ in (3.15) gewichten die Terme der Nebenbedingungen. Sie werden später bestimmt und angepasst, um die Erfüllung aller Nebenbedingungen sicher zu stellen.

Diese Modellierung beinhaltet nun nur noch maximal quadratische Terme in den Problemvariablen x, p, sowie q.

Die QUBO Matrix $Q \in \mathbb{R}^{(I+J+K)}$ mit dem Problemvektor $v = ((p_f)_i, (x_s)_j, (q_{f,s})_k)$ kann nun an Hand der Terme der Kostenfunktion aufgestellt werden. Die Lösung des Problems $\min_x x^T Q x$ gibt dann den Vektor, mit dem die Kostenfunktion minimiert werden kann.

Alle konstanten Terme werden separat summiert, um die gefundenen Lösungen anschließend bewerten zu können.

3.5 Beispiel

Um die Komplexität des Problems zu verdeutlichen, wird für die Simulation ein kleines Beispiel betrachtet:

- Die Menge an Filialen bestehe aus $\mathcal{F} = \{\text{Filiale A}, \text{Filiale B}\}$.
- Die Mischpalette setzt sich aus den 2 beispielhaften Sorten $\mathcal{S} = \{\text{Vanille}, \text{Erdbeere}\}$ zusammen.
- Eine Palette hat 4 Plätze, die beliebig belegt werden können.

Eine triviale Absatzmatrix ist in Tabelle 3.1 definiert.

Tabelle 3.1: Beispielhafte Absatzmatrix $N_{f,s}$ mit trivialer Lösung.

	Vanille	Erdbeere
Filiale A	3	1
Filiale B	3	1

Per Konstruktion wurde ein Beispiel gewählt, bei dem die optimale Konfiguration leicht ersichtlich ist: $x_{\text{Vanille}} = 3$, $x_{\text{Erdbeere}} = 1$. Beide Filialen werden mit je $p_f = 1$ Palette beliefert, um den Bedarf zu decken. Die optimale Lösung ist per Konstruktion bei den Kosten $E(p_f, x_s) = 0$ erreicht.

Für die Codierung der Variable x_s, welche die Bestückung einer Sorte s angibt, werden $J = 5$ Qubits benötigt, um die möglichen Werte von $\{0, 1, 2, 3, 4\}$ darstellen zu können. Die Problemvariablen p_f werden durch $I = 3$ Qubits für den Wertebereich $\{0, 1, 2\}$ Paletten Liefermenge codiert.

Daraus ergibt sich, dass zur Separierung der höheren Terme $I \cdot J = K = 15$ Hilfsqubits q_k für jede Kombination aus Filiale und Sorte benötigt werden. Insgesamt benötigt man also $F \cdot I + S \cdot J + F \cdot S \cdot K = 76$ Qubits, um den Problemvektor für dieses kleine Beispiel repräsentieren zu können.

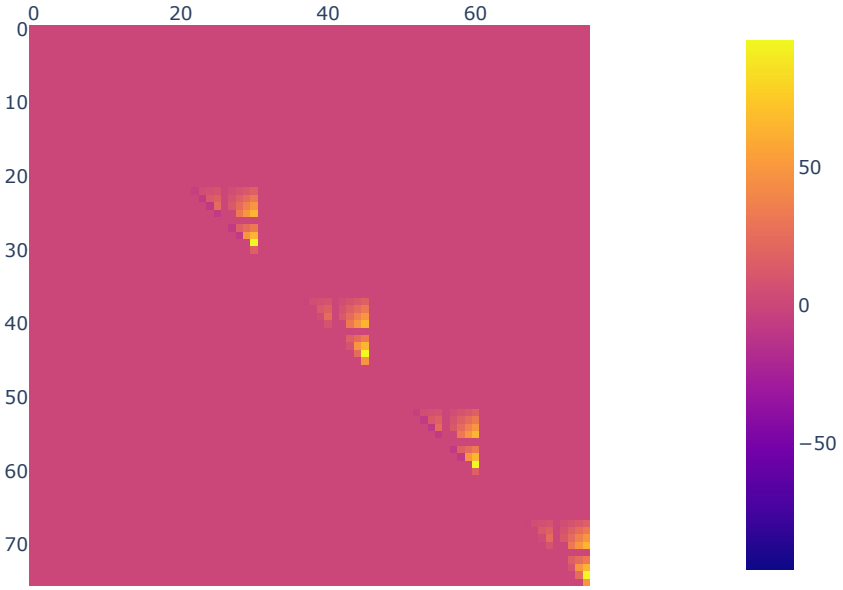

Abbildung 3.1: QUBO Matrix für die Absatzmodellierung.

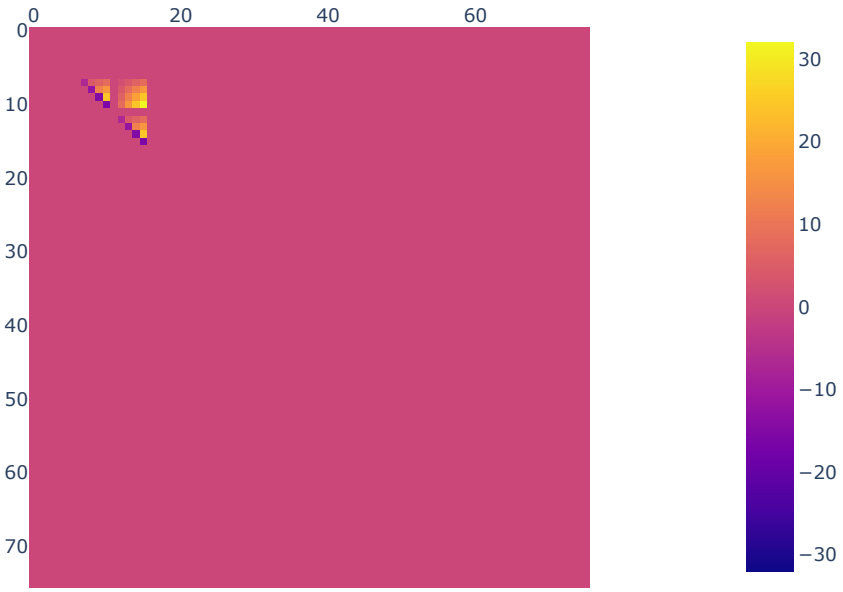

Abbildung 3.2: QUBO Matrix für die Nebenbedingung der Palettengröße.

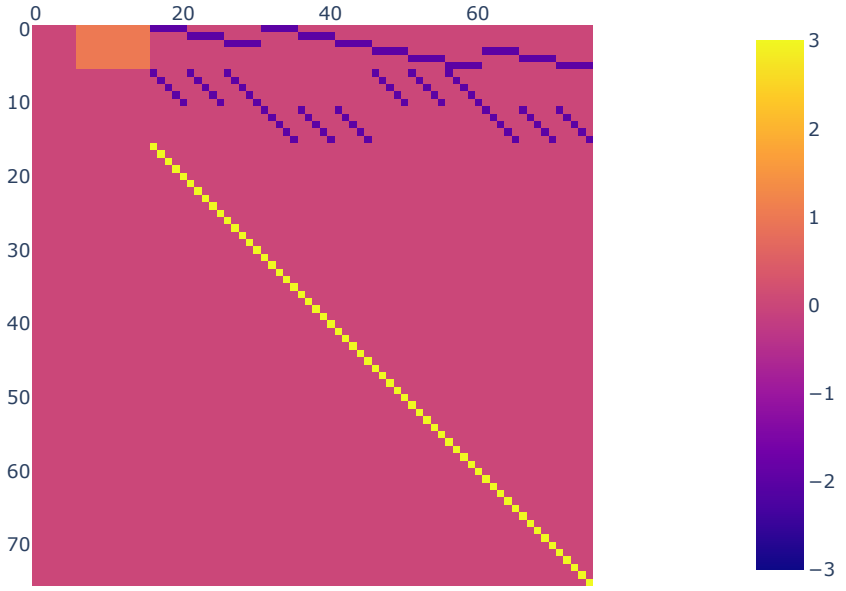

Abbildung 3.3: QUBO Matrix für die Separationsbedingung für q_k.

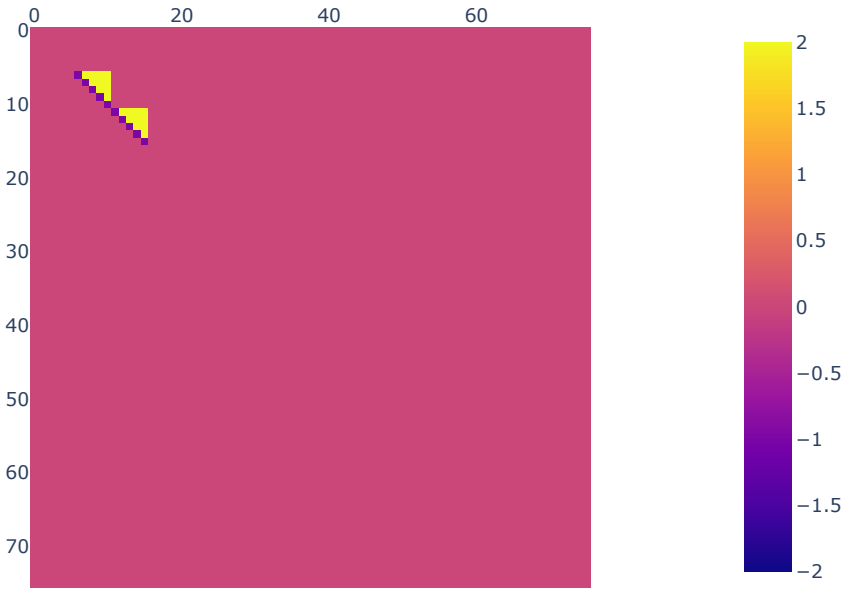

Abbildung 3.4: QUBO Matrix für das One-Hot Encoding für x_s.

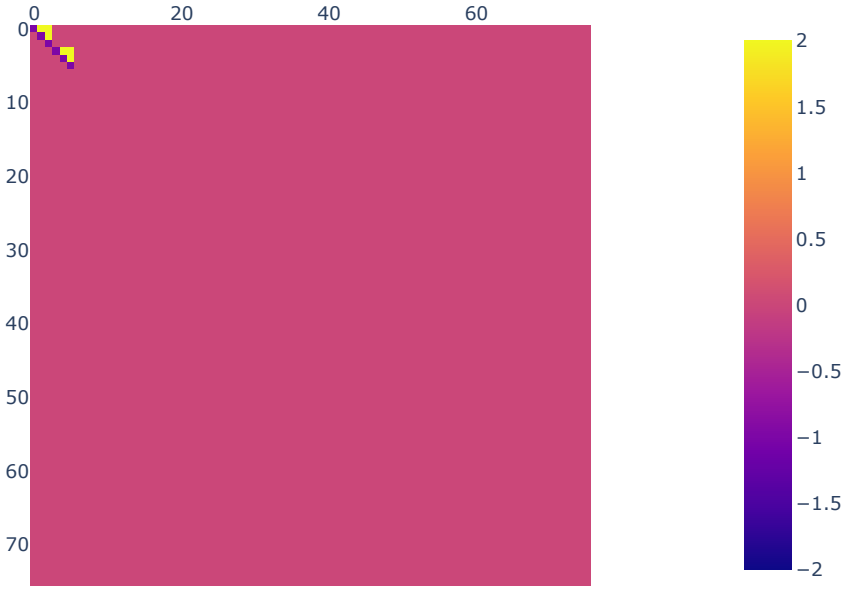

Abbildung 3.5: QUBO Matrix für das One-Hot Encoding für p_f.

Abschließend werden die einzelnen Terme für die verschiedenen Bedingungen aufaddiert zu dem Gesamt-Hamiltonian. Hier findet jeweils eine Gewichtung mit dem zugehörigen Lagrange Parameter λ statt (siehe Gleichung (3.16)).

Ebenso werden auch die konstanten Terme (die sog. offset energy) gewichtet und addiert. Die offset energy ist für die Lage des optimalen Lösungsvektors irrelevant, aber hilfreich für die abschließende Bewertung der gefundenen Lösungen, da die minimale Energie dann nicht kleiner 0 sein kann.

$$H_{\mathrm{P}} = H_{\mathrm{Kosten}} + \lambda H_{\mathrm{NB}} + \lambda_q H_{\mathrm{q}} + \lambda_x H_{\text{One-Hot}-x} + \lambda_p H_{\text{One-Hot }-p} \tag{3.16}$$

Die einzelnen Komponenten des gesamten Hamiltonians für dieses Beispiel sind in Abb. 3.1 bis Abb. 3.5 als QUBO Matrizen dargestellt. Analog zur Addition der Hamiltonians, können auch die QUBO Matrizen gewichtet addiert werden, um das Gesamtproblem zu konstruieren. Im Abb. 3.6 wird beispielhaft die Gewichtung mit den Lagrange Parametern $\lambda = \lambda_q = \lambda_x = \lambda_p = 10$ gezeigt.

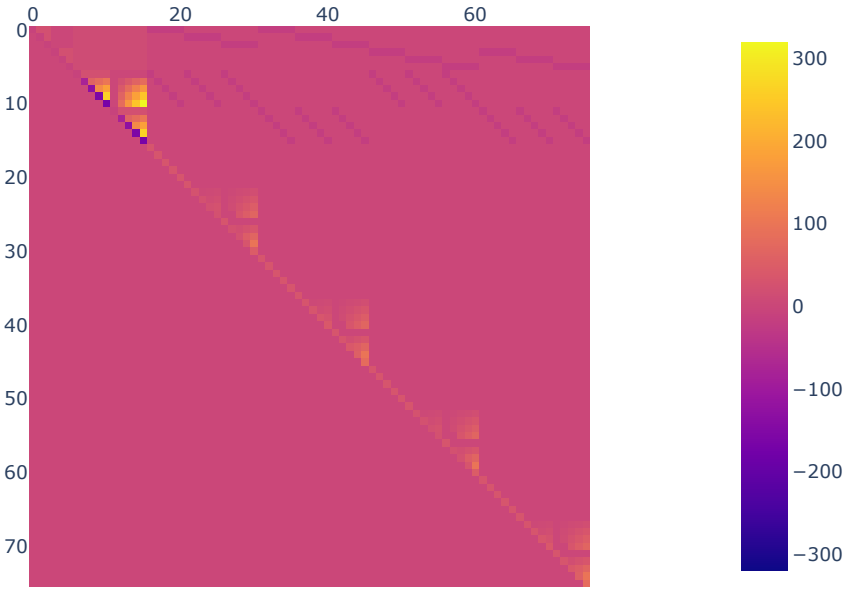

Abbildung 3.6: Gesamte QUBO Matrix für die Abverkaufsmodellierung in 2 Filialen.

3.6 Simulation

Das als QUBO formulierte Problem wurde mit mehreren Methoden gelöst und lieferte folgende Ergebnisse:

- Tabu-Sampler (D-Wave, Heuristischer Algorithmus) [8]

  Der Tabu-Sampler basiert auf einem heuristischen Algorithmus, der ein gewisses Maß an Backtracking betreibt, um gegebenenfalls lokalen Extremstellen zu entweichen [17]. Die Lösung für das obige triviale Problem konnte mit diesem Algorithmus

sicher gefunden werden. Leicht abweichende oder größere Probleme, wie zum Beispiel eine weitere Filiale, konnten nicht mehr gelöst werden.

- Simulated Annealing Sampler (D-Wave, Monte-Carlo-Simulation)

  Beim Simulated Annealing wird eine Metropolis-Hastings Simulation von Bit-Flips durchgeführt, in der sukzessive die Wahrscheinlichkeit („Temperatur“) für das Akzeptieren höher-energetischer Zustände reduziert wird. [14] Für das triviale Beispiel wurde neben einigen nicht optimalen Lösungen auch die optimale Lösung gefunden.

- Leap Hybrid Sampler (D-Wave, Klassisch und QA) [12]

  Der Leap Hyrid Sampler ist ein Service von D-Wave, bei dem klassische Algorithmen mit Quanten Annealing für Teilprobleme kombiniert werden, um so Probleme mit bis zu 1 Millionen Variablen lösen zu können.

  Mithilfe des Leap Hybrid Samplers konnten die optimale Lösung für dieses Problem ebenfalls gefunden werden.

- Quanten Annealer (D-Wave `Advantage_system5.4`, QA)

  Die benötigten 76 Qubits für dieses triviale Problem können auf dem 5000+ Qubit Pegasus Graphen des D-Wave `Advantage_system5.4` [16] mithilfe des Embedding-Composite aus dem D-Wave Ocean Software Development Kit (SDK) eingebettet werden. Dafür werden 804 physische Qubits in Ketten von bis zu 17 Qubits jeweils zu logischen Qubits zusammengefasst.

  Es ließen sich - trotz vielfachem Anpassen der Faktoren für die Strafterme - keine validen Lösungen finden, die alle Nebenbedingungen gleichzeitig erfüllen. Die Vermutung liegt nahe, dass aufgrund der ursprünglich höher als quadratischen Struktur des Problems, die begrenzte Auflösung der elektromagnetischen Felder das Finden von validen Lösungen verhindert. Das Problem muss vor dem Lösen auf die maximalen auftretenden Terme skaliert werden, sodass die schwachen Terme nicht mehr dargestellt werden können.

Aufgrund der hohen Anforderung an die Anzahl und die Konnektivität der Qubits für dieses kleine Problem sowie der schlechten Skalierbarkeit auf mehr Filialen bzw. Sorten, wurde dieser Ansatz nicht weiter verfolgt.

3.7 Betrachtung der Skalierbarkeit

Dieses minimale Modell einer Mischpalette benötigt bereits 76 logische Qubits in dieser Formulierung. Die Berechnung auf echter Quanten-Hardware erfordert wiederum mehr (hier 804) physische Qubits aufgrund der eingeschränkten Konnektivität.

Damit Berechnungen auf der Skala von Business relevanten Größenordungen angestellt werden können, werden weitaus mehr Qubits benötigt als heute Verfügbar sind:

Angenommen die Mischpalette bietet 20 Plätze für Produkte aus 4 Sorten und es wird eine repräsentative Stichprobe von ca. 100 Filialen mit wöchentlichen Liefermengen von 10 Paletten betrachtet, dann benötigt diese Modellierung inklusive Hilfsqubits mindestens $100 \cdot 11 + 4 \cdot 21 + 100 \cdot 4 \cdot 21 \cdot 11 = 90584$ logische Qubits. Hinzu kommt die eingeschränkte Konnektivität, welche die Anzahl der notwendigen physikalischen Qubits nochmal um ein Vielfaches erhöht. Das ist weit über den ca. 5000 physikalischen Qubits, welche die heutige Hardware bieten kann.

4 Anwendungsfall: Filial-Klassifizierung

Bei der Filial-Klassifizierung werden Filialen, welche in ihren Eigenschaften ähnlich sind, in gemeinsame Gruppen eingeteilt. Das bietet den Vorteil, dass Entscheidungen präziser auf die betroffenen Filialen abgestimmt werden können. Im Forecasting Bereich wäre dies zum Beispiel die Entscheidung, dass an Karneval Liefermengen von bestimmten Artikeln angehoben werden für alle Filialen, die an diesem Brauchtumstag ein verändertes Kaufverhalten beobachten.

Filialen und das Kaufverhalten der Kunden ändern sich fortlaufend, so dass diese Klassifizierung regelmäßig überprüft und aktualisiert werden muss. Zudem ist eine Präzisierung der Zuordnung für die Business Entscheidungen von großer Bedeutung.

Kann man also ein optimiertes/kleineres Set an Kennzahlen definieren, anhand derer man die Klassifizierung präziser oder schneller durchführen kann, hat dies zur Folge, dass Entscheidungen effizienter und wirtschaftlich getroffen werden können.

4.1 Mutual Information QUBO

Jede Kennzahl (Feature) des Datensatzes über die Filialen enthält ein gewisses Maß an Information über die bestehende Zuordnung (Target) dieser Filiale. Das ist die gemeinsame Information zwischen Feature und Target. Vergrößert man das genutzte Set an Features um ein Weiteres, hat man zwar mehr Information zum Target, allerdings kann sich diese mit den bisherigen Features zum Teil überschneiden. Gegebenenfalls würde ein anderes Feature, welches an sich weniger Information zum Target besitzt, dennoch einen größeren Zuwachs an „neuer“ Information beitragen.

Es handelt sich also um ein kombinatorisches Problem, bei dem die Aufgabe ist, die Kombination von k Features zu finden, welche gemeinsam die maximale Information zum Target beisteuern.

Man kann die QUBO Matrix Q wie folgt definieren:

$$Q_{ij} = \begin{cases} -I(X_i; Y|X_j) & i \neq j \\ -I(X_i; Y) & i = j \end{cases} \tag{4.1}$$

Sie gibt also den Zugewinn an Information über das Target, wenn Feature X_i zusätzlich zum bereits ausgesuchten Feature X_j selektiert wird, auf den Nebendiagonalen an und auf der Hauptdiagonalen steht der Zugewinn an Information zum Target, wenn nur das Feature X_i gewählt wird.

Um zu vermeiden, dass der Lösungsvektor x_i die Triviale Lösung $x_i = 1 \forall i$ annimmt, wird ein Strafterm hinzugefügt, der die Auswahl auf k Features beschränkt.

$$\lambda(\sum_{i=0}^{N-1} x_i - k)^2 \tag{4.2}$$

4.2 Berechnung der (bedingten) gemeinsamen Information

Auf Grund des Laufzeitvorteils der Berechnung wurde hier der Ansatz der Diskretisierung durch endliche Anzahl von Bins gewählt. Hierbei können selbst Tausende von Datenpunkten innerhalb weniger Sekunden verarbeitet werden. Die Anzahl der genutzten Bins wurde nach [5] durch $D = \lfloor \sqrt{n/5} \rfloor$ abgeschätzt.

Die angenäherte Wahrscheinlichkeitsverteilung $P_X(x)$ wird dann durch die Häufigkeit des Auftretens eines Wertes innerhalb eines Bins angegeben. Analog können die Wahrscheinlichkeitsverteilungen für $P_{(X,Y)}(x, y)$, sowie $P_{(X,Y,Z)}(x, y, z)$ bestimmt werden.

Wie in Abschnitt 2.6 beschrieben, ergeben sich dann die Werte für die (bedingte) gemeinsame Information zwischen den Features des Datensatzes und dem Target Feature. Abschließend kann die Matrix für das MIQUBO Problem aufgestellt werden.

4.3 Datensätze für Klassifizierungsprobleme

Es wurden zwei Datensätze mit nicht-synthetischen Daten untersucht. Der *covertype* [3] Datensatz bietet wenige Features, dafür aber viele Samples für ein Multi-Class Klassifizierungsproblem. Dem gegenüber ist der *scenes* [4] Datensatz ein binäres Problem (bezogen auf das Class-Label „Urban“) mit weniger Samples, dafür aber mehr Features.

Tabelle 4.1: Untersuchte Datensätze für Klassifizierungsprobleme.

	covertype	*scenes*
Samples	581012	2407
Features	12	294
Typ	multi-class	binär
Anzahl an Klassen	7	2
OpenML ID	`1596`	`312`

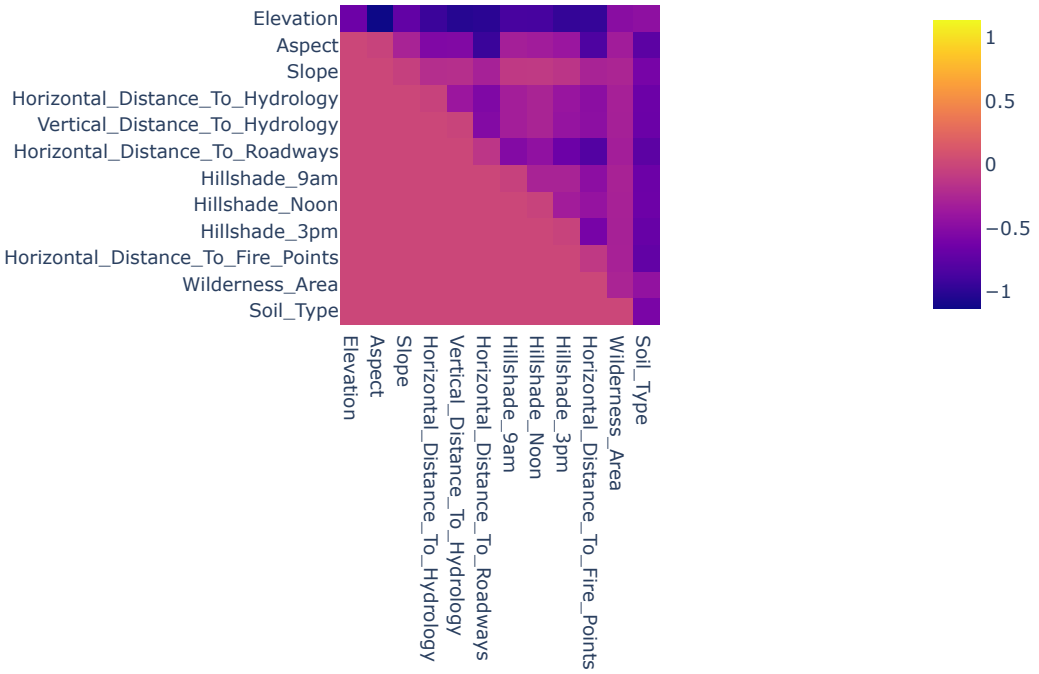

Abbildung 4.1: Mutual information QUBO Matrix für den *covertype* Datensatz mit 12 Features.

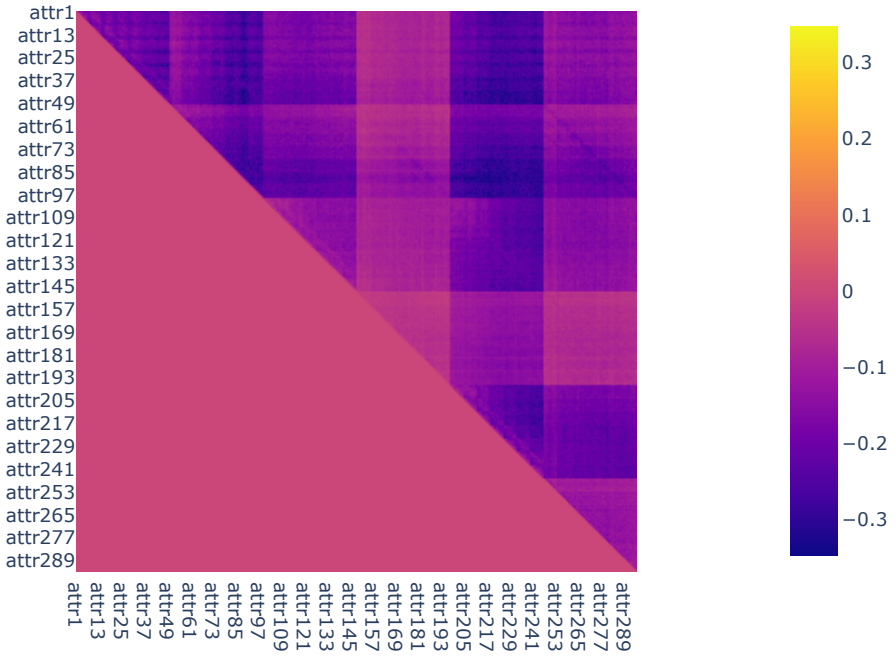

Abbildung 4.2: Mutual information QUBO Matrix für den *scenes* Datensatz mit 294 Features.

Für beide Datensätze wurden die (bedingten-) gemeinsamen Informationsinhalte über die in Abschnitt 2.6 beschriebene Diskretisierung berechnet und die QUBO Matrix aufgestellt. Anschließend wurde das QUBO für alle möglichen Selektionsgrößen $k \in \{0, ..., N\}$ mithilfe von heuristischen Algorithmen, sowie QA gelöst.

Im Falle des QA wurden sowohl das automatische *EmbeddingCoposite* als auch das auf Konnektivität optimierte *CliqueEmbedding* verwendet, um das Problem auf dem QPU-Chip des D-Wave `Advantage_system5.4` zu positionieren.

Dem gegenüber gestellt wurde ein „Greedy"-Algorithmus, der jeweils die k Features mit der höchsten Information über das Zielfeature auswählt.

In Abb. 4.3 bis Abb. 4.6 werden die Feature-Selektionen der verschiedenen Lösungsmethoden für alle Größen von k dargestellt. Aufgrund der Übersichtlichkeit wurden diese Grafiken nur für den *covertype* Datensatz erstellt, da die Anzahl der Features hier noch überschaubar ist.

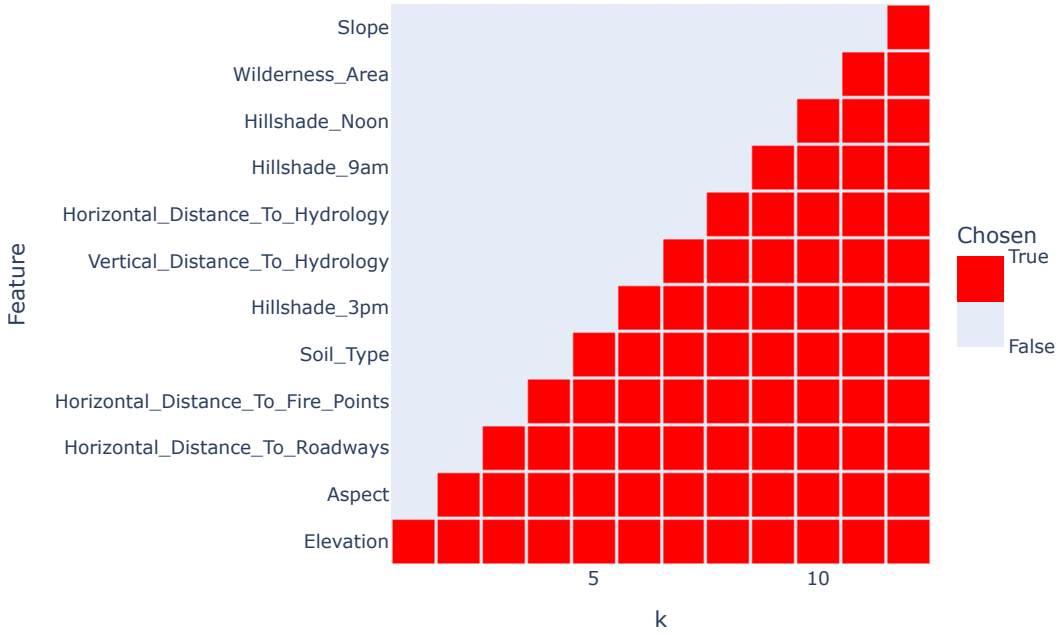

Abbildung 4.3: Selektion an Features für alle k für die exakte Lösung des QUBO, ebenso *Tabu* und *Hybrid*.

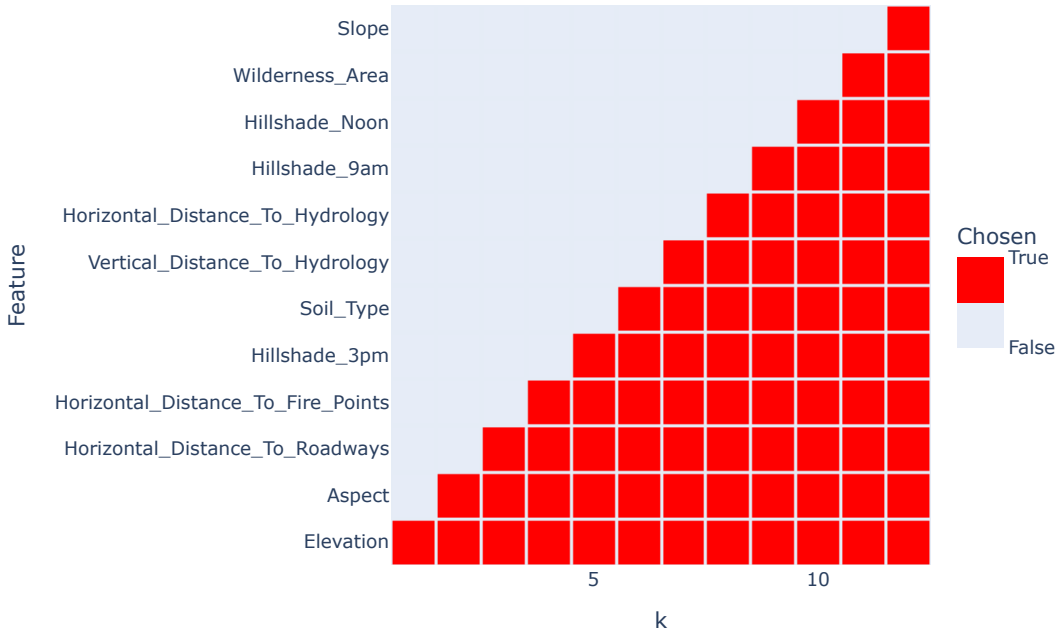

Abbildung 4.4: Selektion an Features für alle k gelöst mit *EmbeddingCompostite* auf dem D-Wave QA.

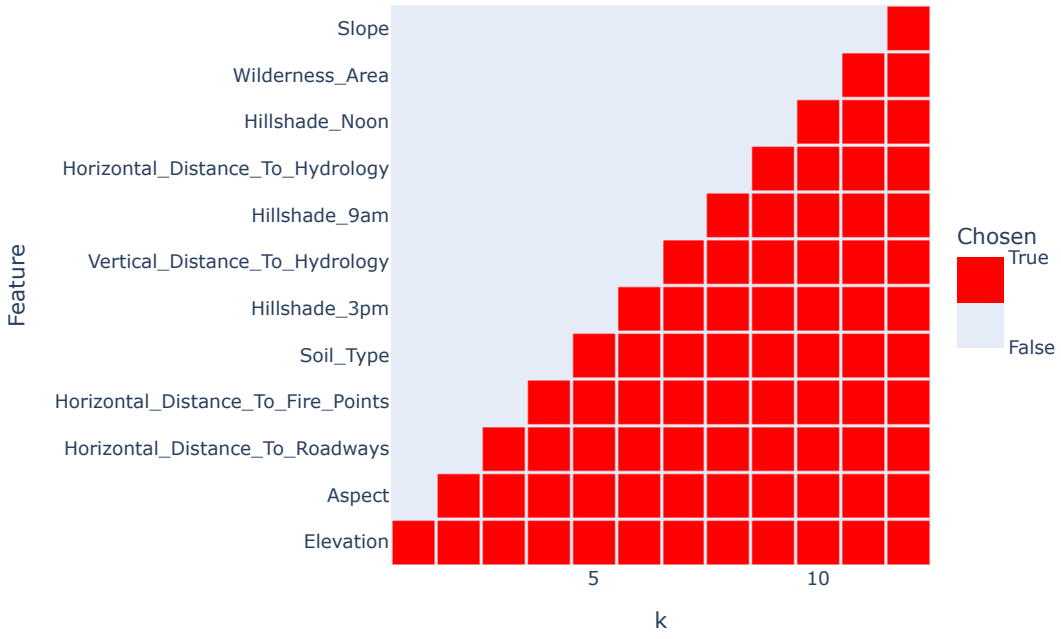

Abbildung 4.5: Selektion an Features für alle k gelöst mit *Clique Embedding* auf dem D-Wave QA.

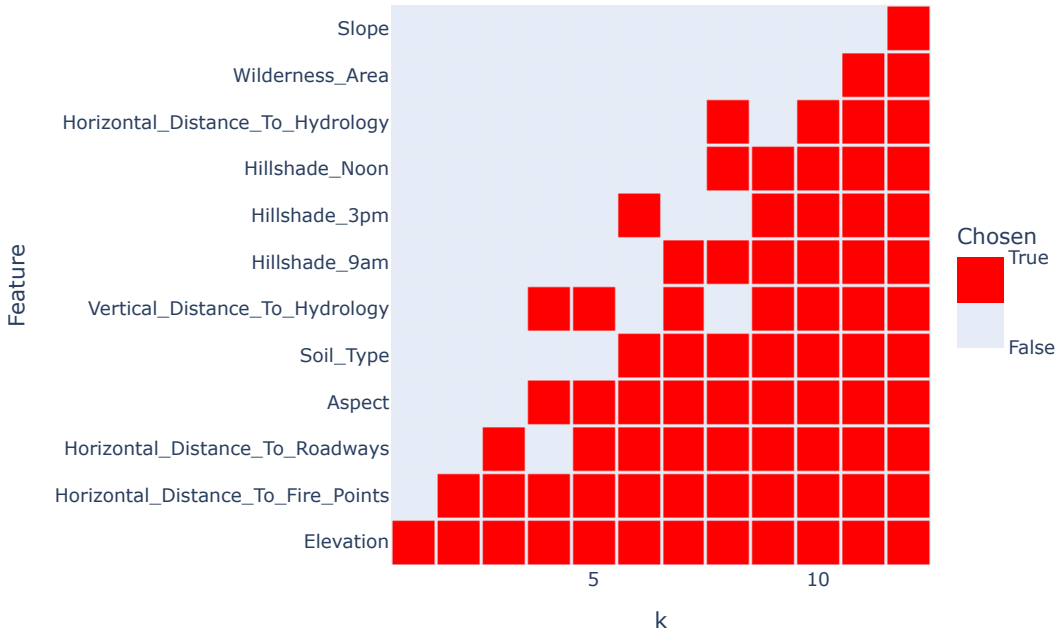

Abbildung 4.6: Selektion an Features für alle k gelöst mit *Simulated Annealing*.

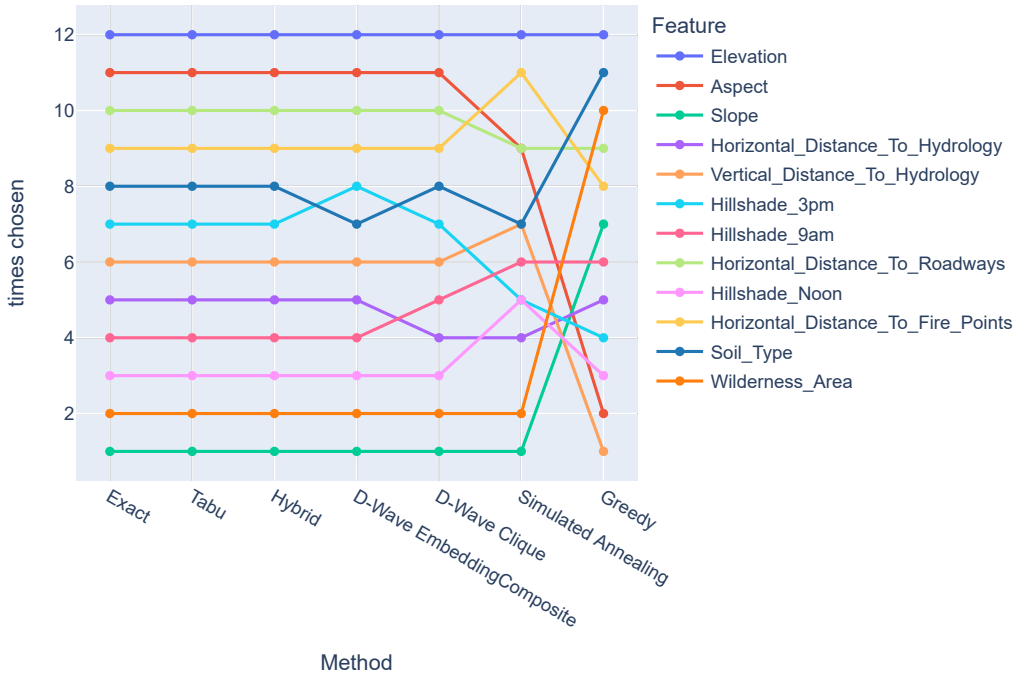

Abbildung 4.7: Relative Feature Importance durch Häufigkeit der Auswahl.

Für den *covertype* Datensatz kann auch eine exakte Lösung für die 12×12 Matrix berechnet werden (siehe Abb. 4.3). Die Methoden *Tabu* und *Hybrid* replizieren genau diese exakte Lösung.

Für den *covertype* Datensatz sind die Auswahlhäufigkeiten der einzelnen Features in Abb. 4.7 gegenübergestellt. Leichte Abweichungen zur Ideal-Selektion treten beim Lösen des QUBO auf dem D-Wave `Advantage_system5.4` auf. Die Selektionen in Abb. 4.4 und 4.5 zeigen aber weiterhin eine gleichmäßige Treppenstruktur, was bedeutet, dass hier nur vereinzelte, paarweise Vertauschungen der Auswahlreihenfolge aufgetreten sind.

Betrachtet man die Selektionen für das *Simulated Annealing* in Abb. 4.6, so fällt auf, dass es hier zum Teil Lücken in der Selektionsreihenfolge gibt. Da die exakte Lösung des QUBOs bekannt ist, wird beim *Simulated Annealing* also häufig eine schlechtere Auswahl getroffen. Durch Erhöhung der Simulationsschritte lässt sich dieses Ergebnis auf Kosten von erhöhter Laufzeit wahrscheinlich verbessern.

Die Lösung des *Greedy* Algorithmus weicht deutlich von allen anderen Methoden ab und wählt Features besonders häufig aus, die von den anderen Methoden nur sehr selten (d.h. erst bei fast vollständigem Featureset) selektiert wurden. Hier wird deutlich, dass das Lösen des kombinatorischen Problems nichttriviale Lösungen erzeugt, die einen größeren Informationsgehalt beinhalten.

4.4 Bewertung durch Kreuzvalidierung

Die Qualität der resultierenden Selektion von k Features wurde zunächst mithilfe eines Random Forrest Classifier (RFC) aus der Bibliothek scikit-learn [18, 1] im Verfahren der Kreuzvalidierung bewertet. Die Methode der k-fachen stratifizierten Kreuzvalidierung führt mit $cv = 5$ geschichteten Stichproben des Datensets (jeweils mit Trainings- und Testdaten) eine Klassifizierung durch. Anschließend wird anhand der Rate von korrekt zugeordneten Samples im Testdatensatz der Erfolg bewertet.

Für die binäre Klassifizierung, wie es beim *scenes* Datensatz der Fall ist, ist dieser Score für sehr wenige Features nicht optimal. Hierzu stelle man sich z.B. einen Datensatz vor, bei dem das Target 80% *Falsch* ist. Ein trivialer Classifier könnte hier pauschal den Wert *Falsch* zuweisen und läge somit bei einem Score von ebenfalls 80%.[21] Daher sind die Bewertungen mit nur einem oder zwei Features kritisch zu betrachten. Der Score gibt dennoch für den Bereich von $k \sim 20\%$ der gesamten Features eine gute Tendenz ob und inwiefern sich das Lösen des Mutual Information QUBO (MIQUBO) lohnt. Schließlich soll die Relevanz der Vorverarbeitung durch diese Feature Selektion untersucht werden.

Der RFC wurde ausgewählt, da er schnell auf einer Vielzahl von Beobachtungen trainiert werden kann. Ein weiterer Vorteil ist, dass er sich auch auf Multi-Label sowie Multi-Class Klassifizierungen erweitern lässt.

Als weiterer Classifier wurde auch der State Vector Classifier (SVC) aus der Bibliothek scikit-learn [18, 2] eingesetzt. Dieser benötigt als zusätzlichen Vorverarbeitungsschritt einen StandardScaler, welcher die Features so transformiert, dass die Datenreihen jeweils Mittelwert 0 und Varianz 1 haben. Es wurde ebenfalls eine Bewertung der Featuresets mittels Kreuzvalidierung durchgeführt. Die Trainingszeit ist bei diesem Classifier erheblich höher als beim RFC, erreicht dafür aber auch i.d.R. einen signifikant höheren success score.

Auch aufgrund dieser langen Laufzeit des SVC wurden vor der Kreuzvalidierung beim *covertype* Datenset die ~ 500.000 Samples auf eine Stichprobe von 10% reduziert.

Besonders für das *scenes* QUBO mit 294 Features ist deutlich zu erkennen (s. Abb. 4.9 & 4.10), dass gegenüber dem *Greedy*-Algorithmus, das Lösen des QUBOs einen Vorteil für die Genauigkeit des Classifiers bringt. In dem Bereich $k \sim 20\%$ hebt sich die höhere Qualität der selektierten Features deutlich vom *Greedy*-Algorithmus ab.

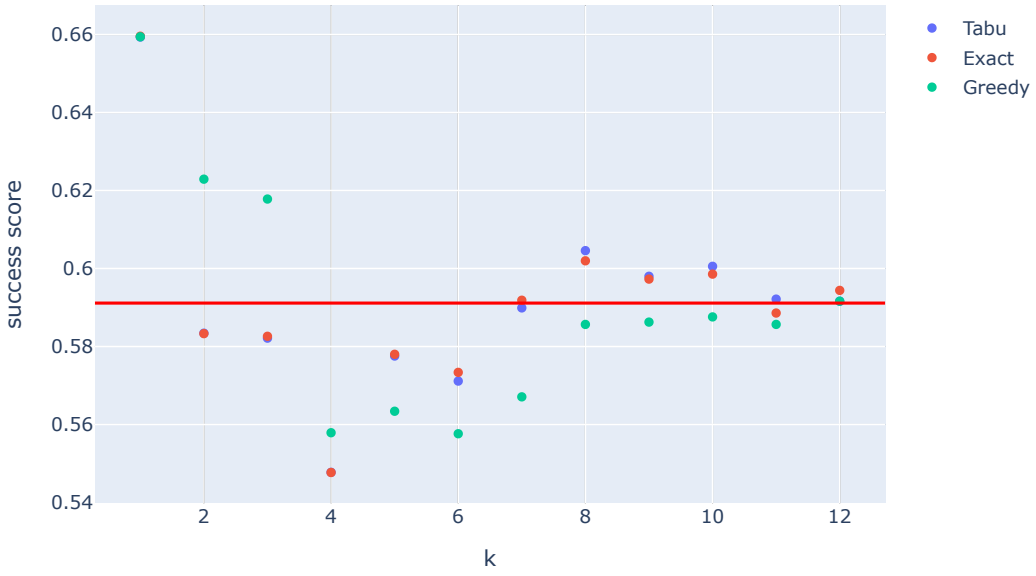

Abbildung 4.8: Success Score bewertet mit RFC für die Lösungen des 12 Feature-QUBOs (*covertype*) mit verschiedenen Algorithmen.

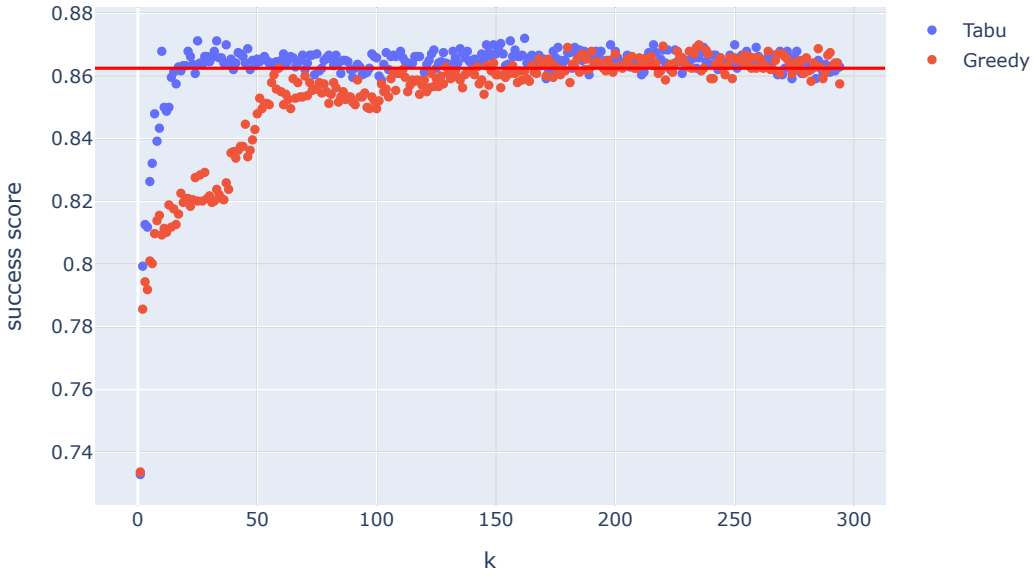

Abbildung 4.9: Success Score bewertet mit RFC für die Lösungen des 294 Feature-QUBOs (*scenes*) mit verschiedenen Algorithmen.

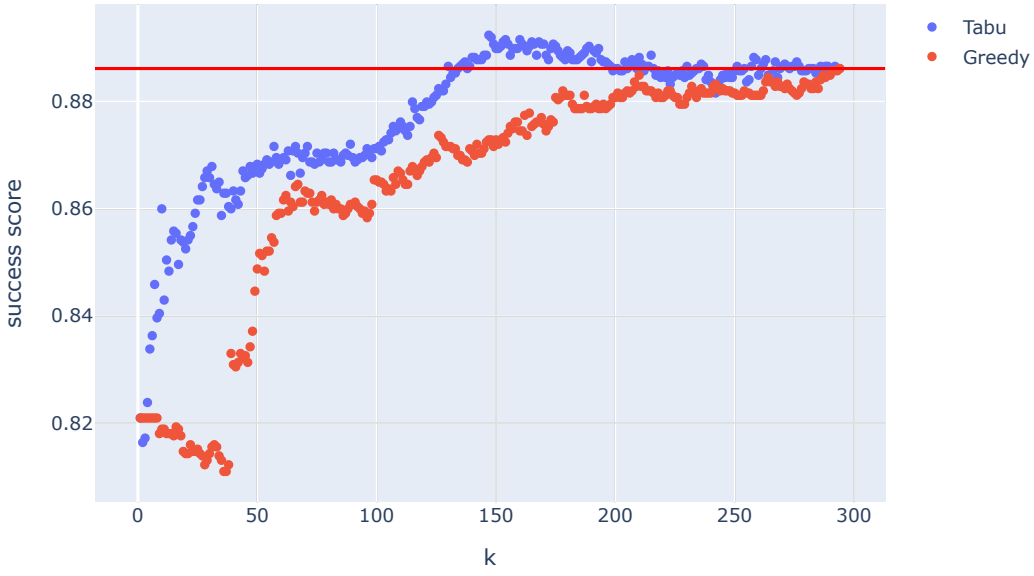

Abbildung 4.10: Success Score bewertet mit SVC für die Lösungen des 294 Feature-QUBOs (*scenes*) mit verschiedenen Algorithmen.

4.5 Fallstudie bei ALDI SÜD

Im Rahmen dieser Arbeit wurde eine Fallstudie mit einem Datensatz der Fa. ALDI SÜD Dienstleistungs-SE & Co. oHG durchgeführt. Das Ziel war es, die Anwendbarkeit der Featureselektion durch gemeinsame Information als Vorverarbeitungsschritt von Daten für die Klassifizierung von Filialen zu testen und zu bewerten. Datenbasis waren 86 Features zu ca. 2000 Filialen, welche im bestehenden Clustering auf 3 Klassen aufgeteilt wurden.

Die (bedingte) gemeinsame Information zwischen den Features wurde erneut mittels Diskretisierung in $D = \lfloor\sqrt{n/5}\rfloor \approx 20$ Bins berechnet. Daraus resultiert die in Abb. 4.11 gezeigte Problemmatrix der Größe 86×86.

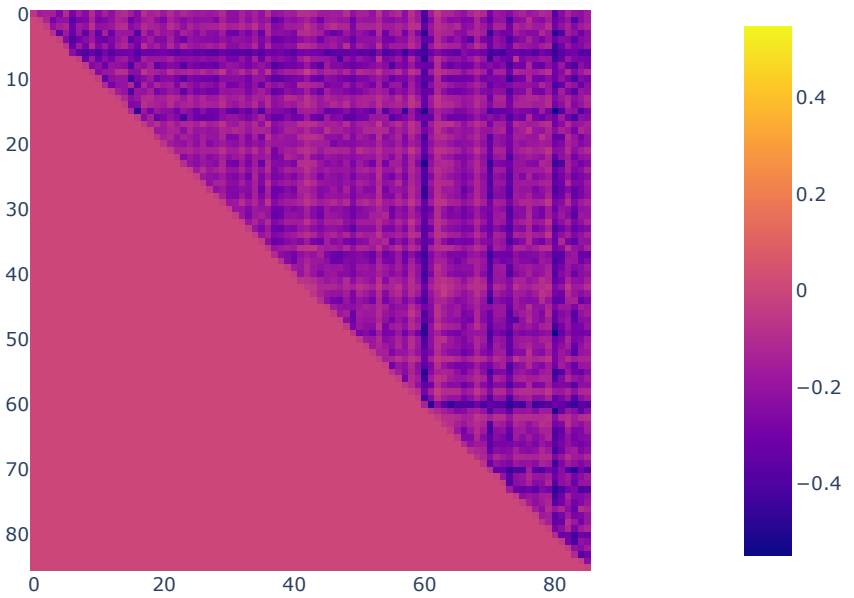

Abbildung 4.11: QUBO Matrix für die Klassifizierung von Filialen anhand von bis zu 86 Features.

4.5.1 Lösung des QUBO

Das QUBO wurde für alle $k \in \{1, ..., N\}$ Größen an Featuresets mit Tabu und Simulated Annealing gelöst. Dem gegenüber wurde wiederum der Greedy Algorithmus gestellt.

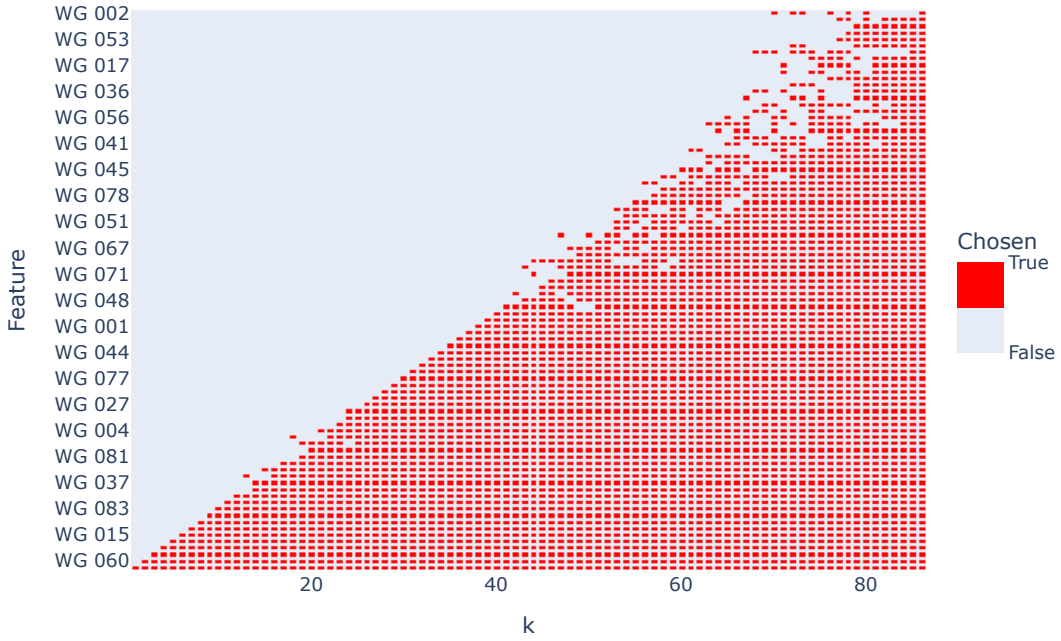

Abbildung 4.12: Lösungen des QUBO für verschiedene Größen von Featuresets mit dem Tabu Algorithmus.

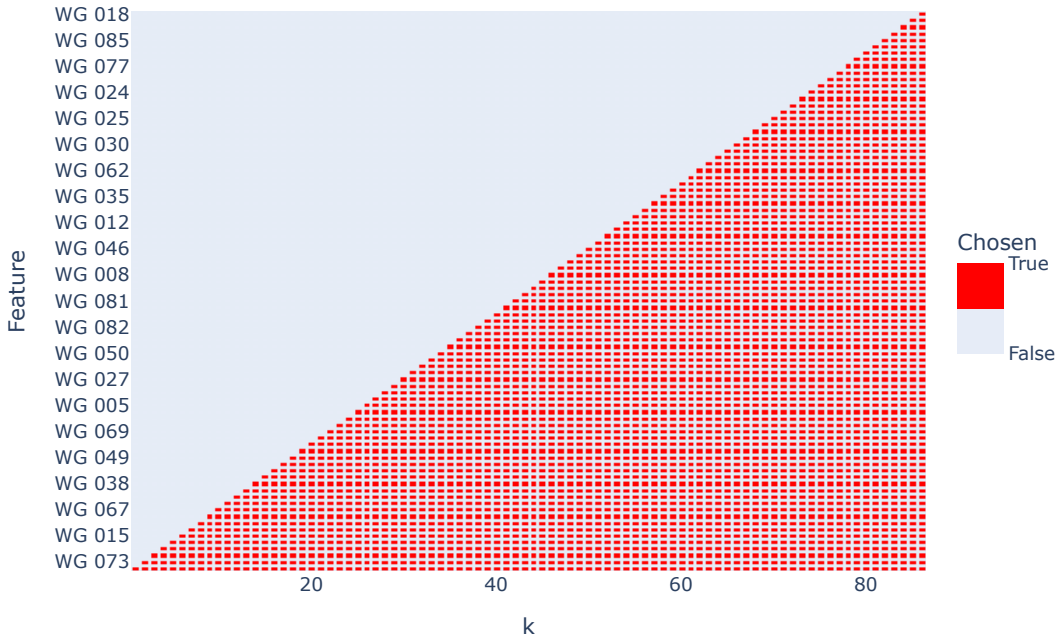

Abbildung 4.13: Lösungen des QUBO für verschiedene Größen von Featuresets mit dem Greedy Algorithmus.

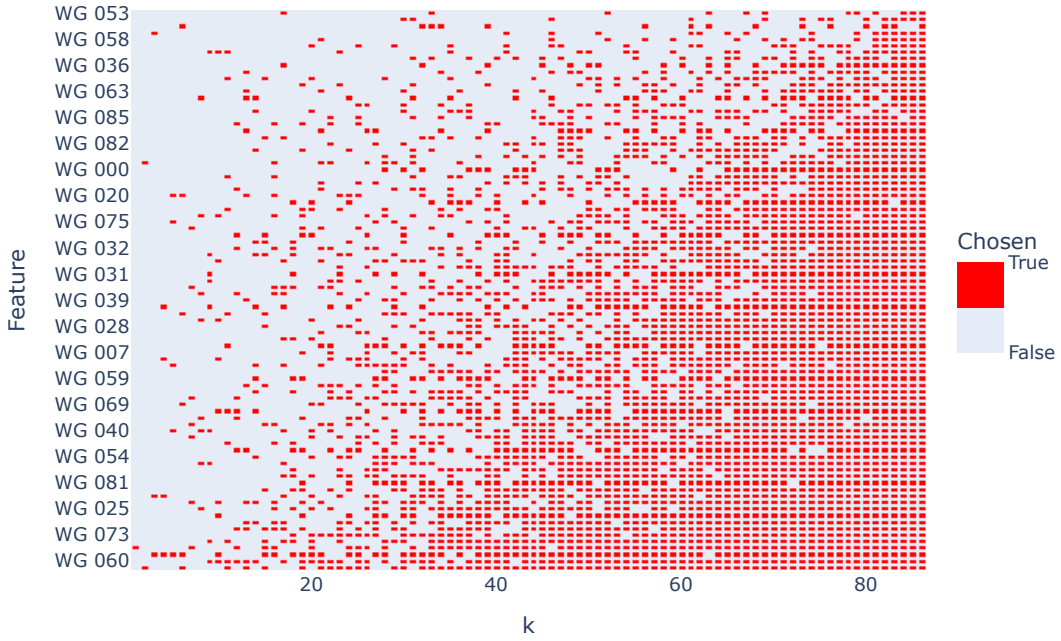

Abbildung 4.14: Lösungen des QUBO für verschiedene Größen von Featuresets mit dem Simulated Annealing Algorithmus.

In den Abbildungen 4.12 bis 4.14 sind für jede Featureset Größe die jeweils selektierten Features farbig markiert. Die Auswahl des *Greedy* Algorithmus (Abb. 4.13) ist ganz klar eine linear steigende Treppe, da jeweils immer das Feature mit dem nächst größeren Informationsgehalt zusätzlich selektiert wird.

Die Auswahl des *Tabu* Samplers (Abb. 4.12) zeigt hier klar, dass es gelegentlich Features gibt, die in einem früheren Schritt ausgewählt wurden, aber danach gegen eine in Summe bessere Kombination von anderen Features ausgetauscht wurden.

Für die Ergebinsse des *Simulated Annealing* (Abb. 4.14) können die Beobachtungen nicht genau bewertet werden: Eine grobe Tendenz zu dem Bild einer aufsteigenden Treppe ist zu erahnen, dennoch wirken die Selektionen zum Teil wahllos und zufällig. Dieses Muster zeigt, dass die Simulation mit den verwendeten Parametern keine guten Ergebnisse liefert. Möglicherweise kann das Ergebnis durch eine Simulation mit vergrößerter Schrittanzahl verbessert werden, was aber auch eine Verlängerung der ohnehin schon langen Laufzeit - verglichen mit den anderen verwendeten Methoden - bedeutet.

4.5.2 Lösung auf dem D-Wave Quantenannealer

Lösungen für das QUBO Problem wurden auch mithilfe des Quantenannealers D-Wave `Advantage_system5.4` gesampled. Da die Struktur des MIQUBO eine All-to-All Konnektivität in den logischen Qubits erfordert, ist es notwendig ein Embedding zu erzeugen, das mehrere physische Qubits zu einem logischen Qubit zusammenschließt. Hierzu wurde zum einen das automatische Embedding (EmbeddingComposite), und zum anderen das speziell auf All-to-All Probleme abgestimmte Clique Embedding aus dem D-Wave Ocean SDK genutzt. Ein mögliches Clique Embedding für dieses QUBO ist in Abb. 4.15 zu sehen.

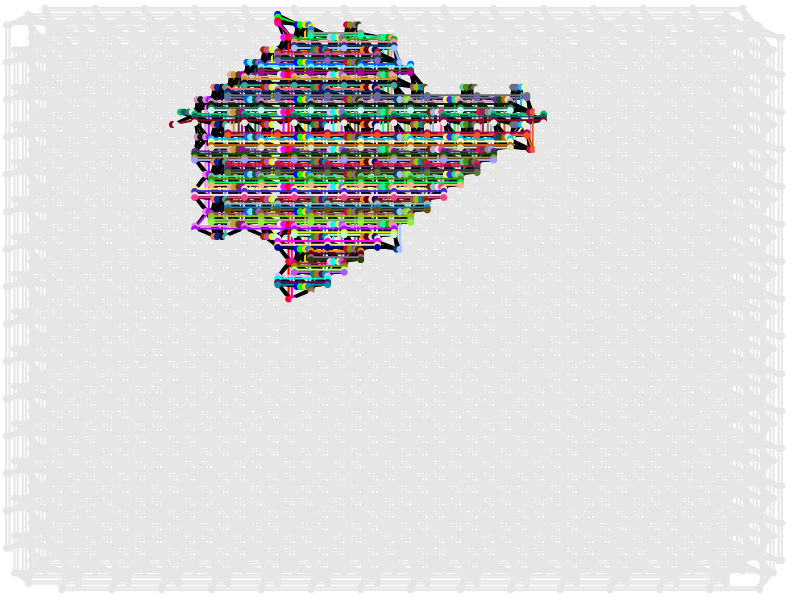

Abbildung 4.15: Clique Embedding für eine 86×86 QUBO Matrix auf dem Pegasus Chip des D-Wave `Advantage_system5.4`.

Auch nach mehrfachem justieren des Parameters für den Strafterm konnten keine zuverlässigen Lösungen gefunden werden, die sowohl die geforderte Anzahl an Features auswählten als auch eine geringe Energie in der Größenordnung der anderen Verfahren aufweisen konnten.

In Abb. 4.16 ist das Sampling Ergebnis für eine manuelle Feinjustage des Strafterms gezeigt. Der Strafterm soll die Selektion hier auf genau $k = 45$ Features einschränken, aber es werden auch viele Lösungen mit mehr oder weniger Features gefunden. Auf der y-Achse wird die Anzahl der tatsächlich selektierten Features für jedes der 500 Samples aufgetragen, auf der x-Achse steht die dazugehörige Energie der Lösung.

Der Parameter $p = 11.2738$ wurde so optimiert, dass das niederenergetischste Sample der Nebenbedingung $k = 45$ entspricht. Im Vergleich findet der Tabu Sampler für das gleiche QUBO genau nur eine Lösung, welche mit Energie $E = -289.97$ deutlich besser ist, als alle Lösungen des D-Wave `Advantage_system5.4`. Der Annealer erreicht im Vergleich

dazu eine minimale Energie von $E = -220.46$. In beiden Fällen wurde die Energie des reinen QUBO ohne den Strafterm berechnet.

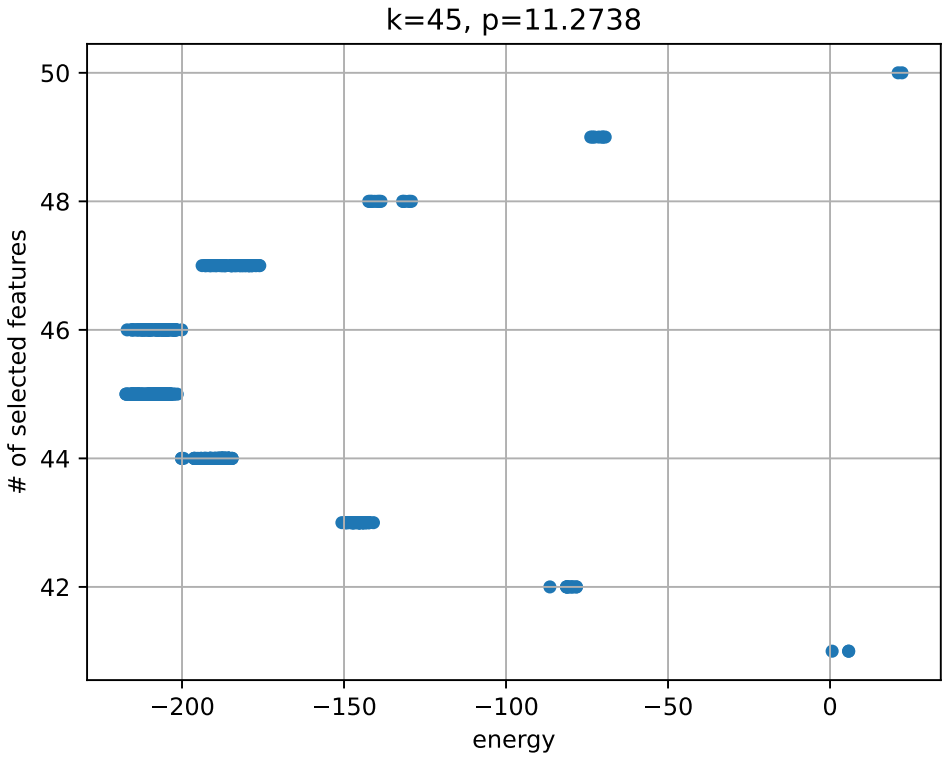

Abbildung 4.16: Samples für $k = 45$ mit Gewichtung $p = 11.2739$ des Strafterms auf dem D-Wave `Advantage_system5.4`.

4.5.3 Bewertung der Selektionen

Die Genauigkeit wurde mithilfe des Kreuzvalidierungsverfahrens anhand der Classifier RFC, SVC sowie ein Histogram Gradient Boosting Classifier (HGB) bestimmt. Bei SVC und HGB wurde zusätzlich eine Skalierung der Featuredaten mit einem StandardScaler aus der Bibliothek sklearns vor der Kreuzvalidierung durchgeführt, um die Datenreihen auf Mittelwert 0 und Varianz 1 zu skalieren.

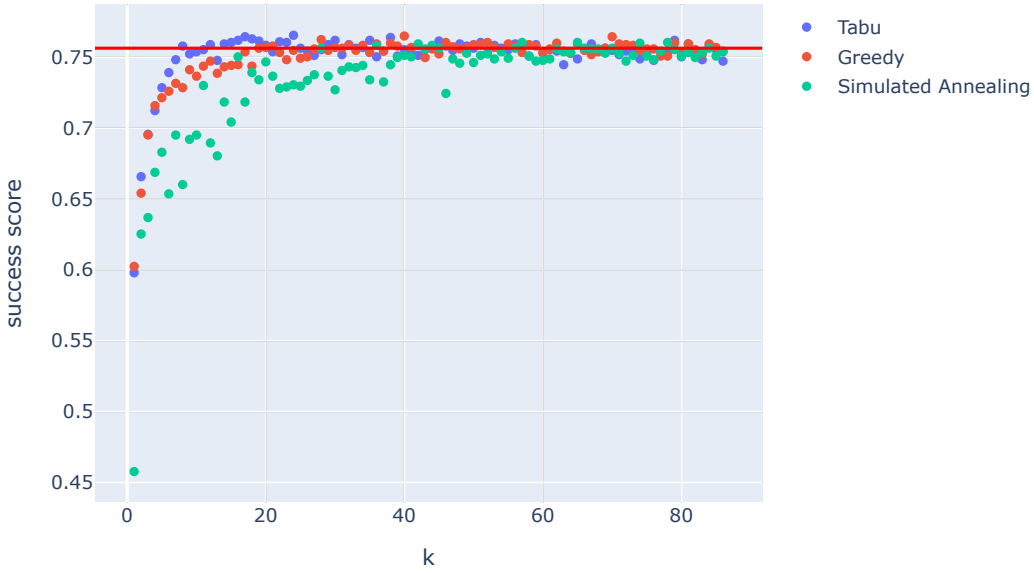

Abbildung 4.17: Kreuzvalidierungsergebnisse des Random Forrest Classifier für verschiedene Lösungsmethoden.

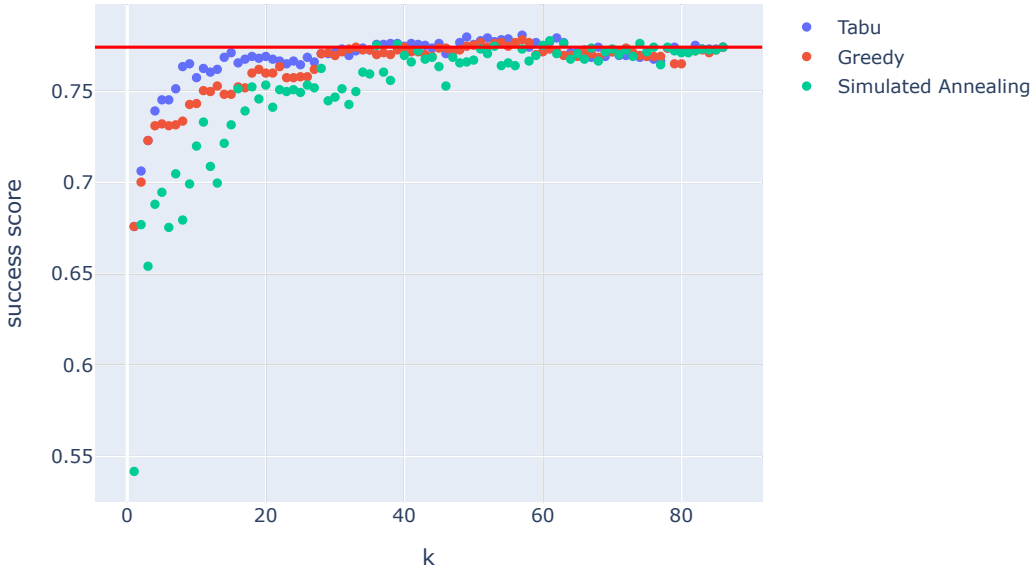

Abbildung 4.18: Kreuzvalidierungsergebnisse des State Vector Classifier für verschiedene Lösungsmethoden.

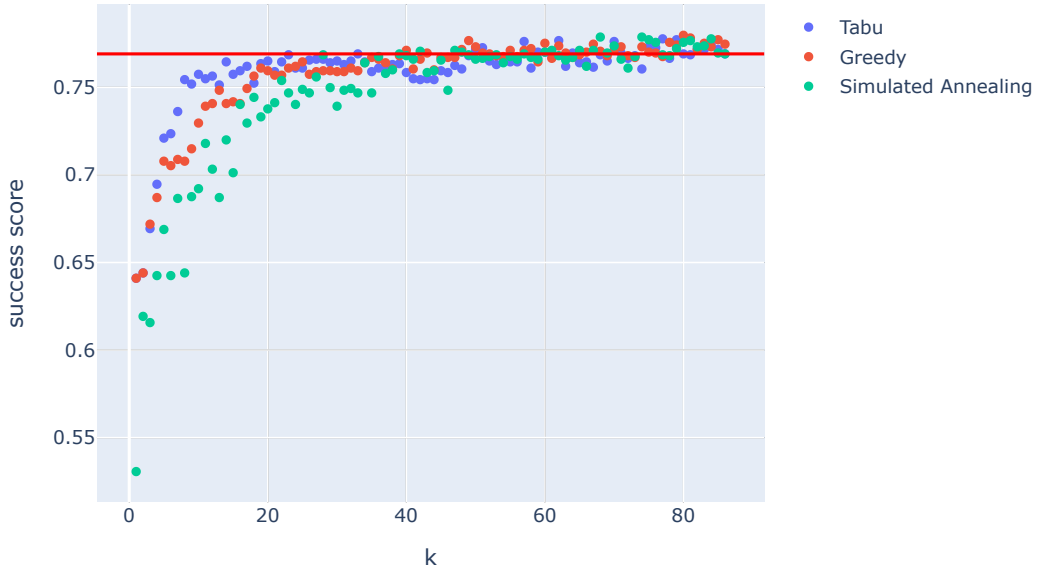

Abbildung 4.19: Kreuzvalidierungsergebnisse des Histogram Gradient Boosting Classifier für verschiedene Lösungsmethoden.

4.5.4 Ergebnisse und Diskussion

Der SVC konnte auf dem Datensatz der Fa. Aldi Süd bestenfalls eine Genauigkeit von etwa 80% für die Klassifizierung erreichen (vlg. Abb. 4.18), und liefert damit die höchste Genauigkeit bei einer Selektion von nur 20% der Features.

Alle Classifier weisen einen ähnlichen Verlauf auf und zeigen, dass die verschiedenen Methoden für die Selektion der Features sich besonders im Bereich von 10 - 20% der Features unterscheiden. Deutlich bessere Werte als alle anderen Methoden erreichen vor allem die Lösungen des Tabu Algorithmus. Das Simulated Annealing schneidet in der Regel als schlechteste Methode ab, welches vermutlich mit den sehr sprunghaften Selektionen zusammenhängt. Durch eine längere Simulation mit mehr Monte-Carlo Schritten ließe sich dieses Ergebnis vermutlich weiter verbessern. Fraglich ist allerdings, ob eine so rechenintensive Methode einen effizienten Lösungsweg bieten kann.

5 Bewertung und Ausblick

In dieser Arbeit wurden zwei Beispiele für Optimierungsprobleme aus dem Supply Chain Management mit Fokus auf den Bereich Bedarfsprognose und Data Science erörtert und auf ihre Lösbarkeit mittels Quanten Computing getestet.

Die Modellierung einer Mischpalette zeigt die grundlegenden Methoden für die Formulierung eines Problems als QUBO. Zudem wird gezeigt, dass selbst triviale Probleme mitunter sehr komplex werden können. Lösungen sind ebenfalls für Spielbeispiele nicht garantiert. Eine hypothetische Skalierung auf eine Business-relevante Größe erzeugt Modelle, die mit den heute verfügbaren QPUs und CPUs faktisch nicht lösbar sind.

Tatsächlich kann das Mutual Information QUBO schon heute den Alltag im Supply Chain Management verändern. Es skaliert viel besser zu den Datenmengen, welche heute für die Entscheidungsfindung verwendet werden. Die Möglichkeit, schnell mit einem optimierten Datensatz Klassifizierungen durchführen zu können und das in manchen Fällen sogar etwas genauer als mit dem gesamten Datensatz, kann einen großen Einfluss auf strategische sowie operative Entscheidungen haben.

Zudem gibt es viel Forschung zu Optimierungsproblemen im Bereich Logistik und Transportketten [19]. Kann hier weiterhin mehr Sichtbarkeit für die Vorteile der neuen Quanten-Technologie mit schnellen und effizienten Lösungen geschaffen werden, wird auch die Entwicklung leistungsfähigerer Quantencomputer vorangetrieben.

Literatur

[1] *1.11. Ensembles: Gradient Boosting, Random Forests, Bagging, Voting, Stacking*. scikit-learn. URL: https://scikit-learn.org/stable/modules/ensemble.html (besucht am 15. 10. 2025).

[2] *1.4. Support Vector Machines*. scikit-learn. URL: https://scikit-learn.org/stable/modules/svm.html (besucht am 15. 10. 2025).

[3] Jock A. Blackard, Dr. Denis J. Dean und Dr. Charles W. Anderson. *Covertype*. UCI Machine Learning Repository, 1998. DOI: 10.24432/C50K5N.

[4] Matthew R. Boutell, Jiebo Luo, Xipeng Shen und Christopher M. Brown. „Learning Multi-Label Scene Classification". In: *Pattern Recognition* 37.9 (Sep. 2004), S. 1757–1771. DOI: 10.1016/j.patcog.2004.03.009.

[5] C. J. Cellucci, A. M. Albano und P. E. Rapp. „Statistical Validation of Mutual Information Calculations: Comparison of Alternative Numerical Algorithms". In: *Physical Review E* 71.6 (22. Juni 2005), S. 066208. DOI: 10.1103/PhysRevE.71.066208.

[6] Nicholas Chancellor. „Domain Wall Encoding of Discrete Variables for Quantum Annealing and QAOA". In: *Quantum Science and Technology* 4.4 (1. Okt. 2019), S. 045004. DOI: 10.1088/2058-9565/ab33c2. arXiv: 1903.05068 [quant-ph].

[7] Thomas M. Cover und Joy A. Thomas. *Elements of Information Theory*. 2nd ed. Hoboken, N.J: Wiley-Interscience, 2006.

[8] *D-Wave Ocean Software Documentation — Ocean Documentation 8.2.0 Documentation*. URL: https://docs.ocean.dwavesys.com/en/latest/index.html (besucht am 15. 10. 2025).

[9] Edward Farhi, Jeffrey Goldstone und Sam Gutmann. *A Quantum Approximate Optimization Algorithm*. Version 1. 14. Nov. 2014. DOI: 10.48550/arXiv.1411.4028. arXiv: 1411.4028 [quant-ph]. URL: http://arxiv.org/abs/1411.4028 (besucht am 15. 10. 2025). Vorveröffentlichung.

[10] Edward Farhi, Jeffrey Goldstone, Sam Gutmann und Michael Sipser. *Quantum Computation by Adiabatic Evolution*. 28. Jan. 2000. DOI: 10.48550/arXiv.quant-ph/0001106. arXiv: quant-ph/0001106. URL: http://arxiv.org/abs/quant-ph/0001106 (besucht am 15. 10. 2025). Vorveröffentlichung.

[11] Fred Glover, Gary Kochenberger und Yu Du. *A Tutorial on Formulating and Using QUBO Models*. 4. Nov. 2019. DOI: 10.48550/arXiv.1811.11538. arXiv: 1811.11538 [cs]. URL: http://arxiv.org/abs/1811.11538 (besucht am 15. 10. 2025). Vorveröffentlichung.

[12] *Hybrid Solvers for Quadratic Optimization - Whitepaper*. URL: https://www.dwavequantum.com/resources/white-paper/hybrid-solvers-for-quadratic-optimization/ (besucht am 15. 10. 2025).

[13] Tadashi Kadowaki und Hidetoshi Nishimori. „Quantum Annealing in the Transverse Ising Model". In: *Physical Review E* 58.5 (1. Nov. 1998), S. 5355–5363. DOI: 10.1103/PhysRevE.58.5355.

[14] S. Kirkpatrick, Jr C. D. Gelatt und M. P. Vecchi. „Optimization by Simulated Annealing". In: *Science* (13. Mai 1983). DOI: 10.1126/science.220.4598.671.

[15] Alexander Kraskov, Harald Stögbauer und Peter Grassberger. „Estimating Mutual Information". In: *Physical Review E* 69.6 (23. Juni 2004), S. 066138. DOI: 10.1103/PhysRevE.69.066138.

[16] Catherine McGeoch und Pau Farre. „The D-Wave Advantage System: An Overview“. In: *D-Wave Technical Report Series 14-1049A-A* (17. Jan. 2022).

[17] Gintaras Palubeckis. „Multistart Tabu Search Strategies for the Unconstrained Binary Quadratic Optimization Problem“. In: *Annals of Operations Research* 131.1 (1. Okt. 2004), S. 259–282. DOI: 10.1023/B:ANOR.0000039522.58036.68.

[18] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot und Édouard Duchesnay. „Scikit-Learn: Machine Learning in Python“. In: *Journal of Machine Learning Research* 12.85 (2011), S. 2825–2830.

[19] Frank Phillipson. *Quantum Computing in Logistics and Supply Chain Management an Overview*. 27. Feb. 2024. DOI: 10.48550/arXiv.2402.17520. arXiv: 2402.17520 [quant-ph]. URL: http://arxiv.org/abs/2402.17520 (besucht am 15. 10. 2025). Vorveröffentlichung.

[20] *QPU-Specific Properties — D-Wave System Documentation Documentation*. URL: https://docs.dwavesys.com/docs/latest/doc_physical_properties.html#advantage-system5-4 (besucht am 15. 10. 2025).

[21] Dennis Willsch, Madita Willsch, Hans De Raedt und Kristel Michielsen. „Support Vector Machines on the D-Wave Quantum Annealer“. In: *Computer Physics Communications* 248 (März 2020), S. 107006. DOI: 10.1016/j.cpc.2019.107006. arXiv: 1906.06283 [cs].

Danksagungen

Ich bedanke mich bei Prof. Dr. rer. nat. Dr.-Ing. Georg Hoever und Dr. Dennis Willsch für die hervorragende fachliche Betreuung während der Erstellung dieser Arbeit.

Ebenso gilt mein Dank dem Jülich Supercomputing Centre für die Bereitstellung von Rechenzeit auf dem D-Wave `Advantage_system5.4` Quantum Annealer in Jülich.

Weiterhin bedanke ich mich auch bei Fa. ALDI SÜD Dienstleistungs-SE & Co. oHG für die Bereitstellung eines anonymisierten Datensatzes, welcher für die Fallstudie des Mutual Information QUBOs untersucht wurde, sowie für die Inspiration zu denen in meiner Arbeit behandelten Problemstellungen aus dem Supply Chain Management.