Lösung des n-Damen Problems auf einem adiabatischen Quantencomputer

Jakov D. Wallbrecher, Jonathan Treffler & Paul Schappert

Gymnasium der regensburger domspatzen | regensburg

# Funktionsweise eines adiabatischen Quantencomputers

## Quantenbit auf einem Quantencomputer

Ein adiabatischer Quantencomputer besteht aus Quantenbits, sogenannten „Qubits“. Diese Qubits stellen das Pendant auf einem Quantencomputer zu den herkömmlichen Bits dar. Im Gegensatz zu ihnen können sich Qubits in einer Superposition befinden, also die Werte 0 und 1 gleichzeitig belegen. Die Qubits werden auch nicht durch einen Transistor dargestellt, wo der Strom an oder aus sein kann, sondern durch den Spin (Englisch für Drehrichtung) eines Elektrons oder der Stromrichtung in einem Supraleitenden Ring.

Da der adiabatische Quantencomputer strenggenommen nur ein Simulator für einen gatebasierten Quantencomputer ist, braucht man immer n² physikalische Qubits um n „echte“ (logische) Qubits zu simulieren. Um dies zu tun, kann man sich die Fähigkeit der physikalischen Qubits zu Nutze machen, dass sie sich durch sogenanntes „verschränken“ mit „couplern“ koppeln lassen. Das bedeutet, dass sie entweder immer den gleichen Wert, oder immer den entgegengesetzten Wert von dem gekoppelten Qubit annehmen. Außerdem kann man ein physikalisches Qubit immer nur mit 4 weiteren gekoppelt werden.

## Chimera-Graph

Der Chimera-Graph beschreibt die Koppelung der physikalischen Qubits. Er zeigt alle aktuell aktive Qubits (Ausfälle sind möglich) und deren Verbindungswege zu anderen Qubits. Auf diese Verbindungswege setzt man nun die coupler. Um ein logisches Qubit zu bekommen, koppelt man mehrere physikalische Qubits mit dem Gleichheits-coupler, um mehr mögliche Verbindungen zu anderen, logischen Qubits zu ermöglichen. Diese kann man dann untereinander so koppeln, dass der Hamiltonian (zu Deutsch Energiefunktion), der das Problem beschreibt, dargestellt wird. Es gibt leider kein „Erfolgsrezept“ um die physikalischen Qubits so zu koppeln, dass sich die entsprechenden logischen Qubits verknüpfen lassen. Daher hat die Firma d-wave eine Library geschrieben, um das embedding (darstellen des Hamiltonians auf dem Chimera-Graphen) zu automatisieren, welche auch wir für unser Problem genutzt haben.

## Hamiltonian und Energieberechnung

|  |  |
| --- | --- |
| a | b |
| c | d |

Der Hamiltonian ist die Basis für den Chimera-Graphen. Er zeigt, in Form einer Matrix, welche Kombination zweier Qubits bestraft oder belohnt werden soll. Wenn man also nicht möchte, dass zwei Damen nebeneinanderstehen (z.B. auf den Feldern a und b) trägt man auf die Position a|b der Matrix den Wert 1 an. Wenn also zwei Damen im Verlauf eines Durchlaufs auf die beiden Felder gesetzt werden würden, würde der Hamiltonian mit dem Wert 1 bestrafen. Dieser Gesamtwert pro Durchlauf heißt Energie.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | a | b | c | d |
| a | 0 | 0 | 0 | 0 |
| b | 1 | 0 | 0 | 0 |
| c | 0 | 0 | 0 | 0 |
| d | 0 | 0 | 0 | 0 |

Diese Energie berechnet der adiabatische Quantencomputer auf Basis der Hamiltonian-Matrix H und eines Vektors q, der die Besetzung der Qubits mit 0 oder 1 darstellt. Um jeden Qubit-wert mit jedem anderen abzugleichen, muss also der Energie-Term folgendermaßen aussehen:

Das ist das gleiche wie:

## Quantum annealing

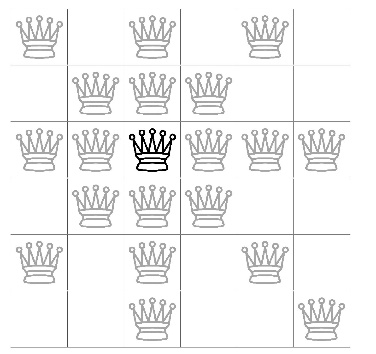
Ein Durchlauf eines Quantum annealing Programms beginnt damit, dass, mittels eines Standard-Hamiltonians, alle Qubits in Superposition „gesetzt“ werden. Langsam wird nun dieser Standard-Hamiltonian schrittweise mit dem eigentlichen Hamiltonian, also dem Chimera-Graph, ausgetauscht.

Die coupler, die durch ein Magnetfeld für jedes Qubit ausgedrückt werden haben eine bestimmte Stärke. Je nachdem, wie das Magnetfeld auf das Qubit einwirkt ist die Wahrscheinlichkeit entweder größer, dass es von der Superposition in die 1, oder dass es in die 0 übergeht. Das Magnetfeld wird immer stärker, bis zum Schluss der Standard-Hamiltonian vollständig gegen den eigenen ausgetauscht ist.

Theoretisch ist es beim unendlich langsamen Übergang vom einen in den anderen Hamiltonian möglich, alle Erhebungen in der Energielandschaft zu tunneln und so in jeder Übergangssituation im besten Energiewert zu bleiben. In der Praxis reichen oft auch wenige Millisekunden, um das Problem zu lösen! Je schwerer das Problem ist, desto öfter „rutscht“ man bei einem Durchlauf in einen schlechteren Zustand.

## Unterschiede zu einem Gate-basierten Quantencomputer

# Das n-Damen Problem

Das n-Damen Problem besteht darin, auf einem Schachbrett n Schachdamen so anzuordnen, dass keine Dame eine andere schlagen kann. Eine Dame darf immer vertikal, horizontal und in alle Richtungen diagonal ziehen. Erstmals formuliert wurde es 1848 von Max Bezzel, allerdings in Zusammenhang mit der Anzahl möglicher Lösungen für die Grundform n=8. Das Problem lässt sich sogar mit brutforce (alle Möglichkeiten durchtesten) in Rechenschritten lösen. Mit heuristischen Optimierungsverfahren lässt sich auch das 1000-Damen Problem in angemessener Zeit lösen. Das besondere an unserem Projekt ist also nicht das Lösen des Problems, sondern das Darstellen als QUBO-Problem (Quadratic Unconstrained Binary Optimaziation) und das Lösen auf einem adiabatischen Quantencomputer. Dieses Problem hat vor uns noch niemand auf einem adiabatischen Quantencomputer gelöst.

# Hamiltonian

Wir haben unseren Hamiltonian (siehe xxx) über einen Term erschlossen. Es muss also eine Formel gefunden werden, die für die bestmögliche Verteilung der Damen den geringsten Wert ausgibt. Diese Formel kann dann als Matrix abgebildet werden. Das folgende Vorgehen haben wir mit unserem selbstgeschriebenen Programm automatisiert.

Eine Schachdame darf vertikal, horizontal und in beide Richtungen diagonal ziehen. Also darf auf jeder Horizontale, Vertikale und Diagonale höchstens eine Dame stehen!

Beginnen wir mit den Horizontalen am Beispiel von n=4:

Im nxn Schachfeld gibt es n Horizontalen, wo jeweils eine Dame stehen muss, bzw. nur eine Dame stehen darf. Der Wert muss also dann am geringsten sein, wenn genau 1 Dame in der Horizontale steht. Dazu addiert man alle Felder in der Horizontalen (0, wenn keine Dame; 1 wenn eine Dame), zieht 1 ab und quadriert dies, damit die Energie bei einer Dame in der Horizontalen kleiner ist, als bei 0 Damen:

Dieses Vorgehen wendet man in jeder Horizontalen an und addiert dies dann.

Bei den Vertikalen ist das Vorgehen im Grunde gleich, da auch in jeder Vertikalen genau 1 Dame stehen muss.

In den Diagonalen darf jeweils eine oder 0 Damen stehen, da es immer Diagonalen in eine Richtung (von links oben nach rechts unten oder von rechts oben nach links unten) gibt, von denen nur auf n eine Dame stehen kann.

Für null und einer Dame auf der Diagonale resultiert also eine Energie von 0,25, bei mehr Damen auch höhere Werte. Für die andere Diagonalrichtung gilt dasselbe. Die einzelnen Eckfelder werden in der Gesamtformel nicht berücksichtigt, weil die Gesamtenergie auf diesen Diagonalen nur 1 oder 0 sein kann, also die Bedingung in jedem Fall erfüllt ist. Hier ein kurzer Einblick in unseren Programmcode, der die Felderkombinationen für alle Diagonalen von links oben nach rechts unten berechnet:

//diagonal obere Hälfte links oben nach rechts unten ohne diagonale

for (int i=1; i<=n-1; i++) {

  ArrayList <PVector> hamiltonianTermIntern = new ArrayList<PVector>();

  for (int x=1, y=i, j=1; (x<=n||y>=1)&&j<=i; x++, y--, j++) {

    hamiltonianTermIntern.add(new PVector(x, y));

  }

  ArrayList <Summand> hamiltonianTermInternAusmult = loeseKlammernAufDiagonal(hamiltonianTermIntern);

  for (Summand s : hamiltonianTermInternAusmult) {

    hamiltonianTerm.add(s);

  }

}

//diagonal untere hälfte links oben nach rechts unten

for (int i=1; i<=n; i++) {

  ArrayList <PVector> hamiltonianTermIntern = new ArrayList<PVector>();

  for (int x=i, y=n, j=1; (x<=n||y>=1)&&j<=n; x++, y--, j++) {

    hamiltonianTermIntern.add(new PVector(x, y));

  }

  ArrayList <Summand> hamiltonianTermInternAusmult=loeseKlammernAufDiagonal(hamiltonianTermIntern);

  for (Summand s : hamiltonianTermInternAusmult) {

    hamiltonianTerm.add(s);

  }

}

Am Beispiel vom 4-Damen-Problem ergibt dies also eine erste Hamiltonian-Formel:

Es kommen hier nur 2 Arten von Termen auf, die sich wie folgt vereinfachen lassen:

Da beim Abbilden auf die Matrix die Quadrate nicht berücksichtigt werden:

Der andere Term für die Bedingung „eine oder 0 Damen“:

Die Konstanten 1 oder 0,25 werden in der Matrix nicht berücksichtigt, da sie eine Verschiebung des Energiegraphs entlang der y-Achse bewirken, welche das Minimum, welches gefunden werden muss, nicht verändert.

Die daraus resultierende Formel der Form Kann nun sehr einfach auf die Hamiltonian-Matrix der Größe übertragen werden. Dazu wird (beispielsweise bei ) auf das Feld (a | k) der Wert 2 geschrieben. Da allerdings alle Werte in den „oberen“ Teil der Matrix geschrieben werden wird der Wert 2 bei (k | a) eingetragen. Bei z.B. -a würde auf Feld (a | a) der Wert -1 geschrieben werden. Daraus resultiert beim 4-Damen-Problem die rechts abgebildete Hamiltonian-Matrix.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p |
| a | -2 | 2 | 2 | 2 | 2 | 2 | 0 | 0 | 2 | 0 | 2 | 0 | 2 | 0 | 0 | 2 |
| b | 0 | -2 | 2 | 2 | 2 | 2 | 2 | 0 | 0 | 2 | 0 | 2 | 0 | 2 | 0 | 0 |
| c | 0 | 0 | -2 | 2 | 0 | 2 | 2 | 2 | 2 | 0 | 2 | 0 | 0 | 0 | 2 | 0 |
| d | 0 | 0 | 0 | -2 | 0 | 0 | 2 | 2 | 0 | 2 | 0 | 2 | 2 | 0 | 0 | 2 |
| e | 0 | 0 | 0 | 0 | -2 | 2 | 2 | 2 | 2 | 2 | 0 | 0 | 2 | 0 | 2 | 0 |
| f | 0 | 0 | 0 | 0 | 0 | -2 | 2 | 2 | 2 | 2 | 2 | 0 | 0 | 2 | 0 | 2 |
| g | 0 | 0 | 0 | 0 | 0 | 0 | -2 | 2 | 0 | 2 | 2 | 2 | 2 | 0 | 2 | 0 |
| h | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -2 | 0 | 0 | 2 | 2 | 0 | 2 | 0 | 2 |
| i | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -2 | 2 | 2 | 2 | 2 | 2 | 0 | 0 |
| j | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -2 | 2 | 2 | 2 | 2 | 2 | 0 |
| k | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -2 | 2 | 0 | 2 | 2 | 2 |
| l | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -2 | 0 | 0 | 2 | 2 |
| m | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -2 | 2 | 2 | 2 |
| n | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -2 | 2 | 2 |
| o | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -2 | 2 |
| p | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -2 |

# Lösung des n-Damen-Problems mit Optimierungsverfahren auf klassischen Computern

Vor dem Ausführen auf einem Quantencomputer haben wir unseren Hamiltonian noch mit mehreren Optimierungsverfahren getestet und diese verglichen. Diese Optimierungsverfahren wollen wir im Folgenden vorstellen.

## Greedy-Algorithmus

Alle Optimierungsverfahren arbeiten mit dem Ansatz, zuerst eine Änderung zu machen (hier: auf einem beliebigen Feld eine Dame wegnehmen/hinzufügen) um dann zu bewerten, ob die Änderung angenommen oder rückgängig gemacht wird. Der einzige Unterschied besteht also in den Kriterien, ob die Änderung angenommen wird.

Der Greedy-Algorithmus (englisch für gierig) nimmt die Änderung nur dann an, wenn die Energie besser ist, als im Zustand vor der Änderung. Das führt zwar bei vielen np-harten Problemen zu einer unzureichenden Lösung, da man in einem lokalen Minimum bleiben könnte. Allerdings hat er, in Verbindung mit unserem Hamiltonian, der ja während des Optimierungsvorgangs beliebig viele Damen zulässt, zu überraschend guten Lösungen geführt. Die Anzahl der notwendigen Durchläufe schwankt allerdings (bei gleicher Ausgangsbesetzung des Schachfelds) sehr stark.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

## Threshold-Accepting

Beim Threshold-Accepting (englisch für Schwellenwertannahme) wird die Änderung dann angenommen, wenn der Energieunterschied zwischen den Energien vor und nach der Änderung unter einem bestimmten Schwellenwert liegt. Dieser Schwellenwert wird nach jeder Änderung etwas herabgesetzt. So dürfen am Anfang noch mehr Situations-verschlechterungen vorkommen, wohingegen später nahezu nur noch Energieverbesserungen angenommen werden. Dies führt dazu, dass ein lokales Minimum in der Energielandschaft überwunden werden kann, um so schneller zu einem besseren Ergebnis zu gelangen. Das Threshold-Accepting gehört, im Gegensatz zum greedy-Algorithmus, zu den heuristischen Optimierungsverfahren.

## Simulated Annealing

Simulated Annealing (englisch für simuliertes Abkühlen) funktioniert sehr ähnlich wie der greedy-Algorithmus. Im Gegensatz zu greedy-Algorithmus wird die Änderung zu einer Wahrscheinlichkeit von auch angenommen, wenn die Energie schlechter ist als vor der Änderung. Der Parameter T ist die sog. Temperatur, welche nach jedem Durchlauf abgesenkt wird (Daher „annealing“, da die Temperatur „abkühlt“). Das führt dazu, dass ein lokales Minimum in der Energielandschaft noch effizienter und wahrscheinlicher überwunden werden kann als bei Threshold-Accepting.

## Great Deluge Algorithmus

Der Great Deluge Algorithmus (englisch für Sintflut Algorithmus) lässt eine Änderung dann zu, wenn die neue Energie unter einem Schwellenwert liegt. Dieser Schwellenwert wird nach jeder Änderung herabgesetzt. Dieser Algorithmus hat mit unserem Hamiltonian sehr schlechte Ergebnisse erreicht. Um eine Lösung zu finden brauchte dieser Algorithmus oft ein Vielfaches der Durchläufe der anderen Optimierungsverfahren um das globale Minimum zu erreichen.

# Auswertung der Ergebnisse vom Quantencomputer

# Danksagung und Ausblick

Wir bedanken uns bei unserem Projektbetreuer René Grünbauer, für die Idee und den Kontakt zum Forschungszentrum Jülich. Außerdem möchten wir uns bei Herrn Dennis Wilsch, der beim Forschungszentrum Jülich im Bereich quantum computing arbeitet, für viele Erklärungen und Hilfestellungen rund um das Benutzen und Steuern des Quantencomputers bedanken. Nicht zuletzt wollen wir xxx für die Rechenzeit auf dem Quantencomputer danken.

Bis zum Präsentationstermin möchten wir noch versuchen, das „knights-tour problem“ (zu Deutsch Rösselsprung-Problem) auf einem Quantencomputer zu lösen. Wir hoffen, das wir dieses weitere Schachproblem auf Basis unseres bisher erworbenen Wissens in kurzer Zeit ebenso lösen können.

# Impressum