Lösung des n-Damenproblems auf einem adiabatischen Quantencomputer

Jakov D. Wallbrecher, Jonathan Treffler & Paul Schappert

Gymnasium der regensburger domspatzen | regensburg

# Projektidee

Auf einem n x n großen Schachfeld sollen n Damen so aufgestellt werden, dass keine Dame eine andere bedroht. Dieses klassische Problem haben wir auf einem echten Quantencomputer gelöst.   
Bei unserem Projekt geht es aber weniger um die eigentliche Lösung des n-Damenproblem (hierfür gibt es bereits viele Algorithmen), sondern um den Weg dieses Problem so aufzubereiten, dass es mit einem "Quantum Annealer" (hierzu später mehr) gelöst werden kann. Unseres Wissens nach ist das n-Damenproblem bisher von niemanden für einen Quanten Annealer adaptiert worden.

Für unser Vorhaben mussten wir das Damenproblem zuerst in die Sprache der Mathematik übersetzen, um dann die Aufgabenstellung auf einen Quantencomputer übertragen zu können. Es galt eine Funktion zu finden, die als Eingabe die einzelnen Positionen der Damen auf dem Schachbrett erhält, wobei alle 2n möglichen Konstellationen als Eingabewert erlaubt sind, also auch solche, die das Problem nicht lösen, wie etwa keine Dame oder mehr als n Damen auf dem Brett und Konstellationen, bei denen mindestens eine Dame eine andere schlagen kann.

Unsere Funktion liefert für jede mögliche Verteilung der n Damen einen Wert, den man in der Optimierung als "Energie" der Verteilung bezeichnet. Die Lösung des n-Damenproblems läuft dann darauf hinaus, eine Konstellation zu finden, bei der die Energiefunktion ein globales Minimum hat. Dies ist genau dann der Fall ist, wenn n Damen auf dem Brett stehen, die sich gegenseitig nicht schlagen können.

Um die Energiefunktion auf Korrektheit zu testen, schrieben wir zuerst Programme, die unsere Funktion mit Optimierungsalgorithmen wie dem Threashold Accepting, Simulated Annealing oder dem Great Deluge Algorithmus auf klassischen Computern minimiert. Anschließend haben wir unser Programm auf einen adiabatischen Quantencomputer der Firma D-Wave übertragen und ausgeführt. Die Rechenzeit auf dem zehn Millionen Dollar teuren Quantencomputer haben wir über das Forschungszentrum in Jülich (Prof. Dr. Kristel Michielsen) erhalten.

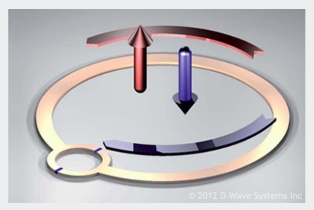
Wie nun so ein adiabatischer Quantencomputers funktioniert, welche Probleme wir lösen mussten, um das Damenproblem auf den Quantencomputer zu übertragen und zu welchen Ergebnissen wir beim Ausführen des Programms gekommen sind, möchten wir im Folgenden genauer erläutern.

*Der D-Wave Quanten Annealer 2000Q, auf dem wir unsere Programme ausgeführt haben*

# Funktionsweise eines adiabatischen Quantencomputers

## Quantenbit auf einem Quantencomputer

Quantencomputer rechnen anders als klassische Computer nicht mit Bits, die den Wert 0 oder 1 annehmen, sondern mit Quantenbits, sogenannten „Qubits“. Im Gegensatz zu klassischen Bits können sich Qubits in einer Überlagerung (quantenmechanisch "Superposition") der Werte 0 und 1 befinden. Erst beim Auslesen des Qubits, was physikalisch einer Messung seines Werts entspricht, wird einer der beiden Zustände mit einer bestimmten Wahrscheinlichkeit angenommen, die von dem Zustand des Qubits vor der Messung abhängt. Ein Qubit befindet sich also quantenmechanisch im Zustand + , wobei a und b (komplexe) Zahlen sind, deren Betrag zwischen kleiner gleich 1 ist und für die gilt: a² + b² = 1. Beim Auslesen (Messen) des Qubits erhält man den Wert 0 mit einer Wahrscheinlichkeit von a² und 1 mit einer Wahrscheinlichkeit von b².



*Qubit auf einem D-Wave*

Qubits werden nicht durch einen Transistor realisiert (Strom an oder aus), sondern z.B. durch den Spin (Drehrichtung) eines Elektrons oder, wie bei dem von uns verwendeten Quantencomputer, durch den Umlaufsinn der Stromrichtung in einem supraleitenden Ring (siehe Abb. rechts). Die Superposition bedeutet also die Fließrichtung des Stroms in beide Richtungen.

Man unterscheidet zwei Arten von Quantencomputer: Universelle Quanten-gattercomputer, bei denen man jedes Qubit mit jedem anderen verrechnen (koppeln) kann und adiabatische Quantencomputer, auch Quantum Annealer genannt. Letztere berechnen die Lösung eines Problems wie folgt:

*"Gemäß den Gesetzen der Quantenmechanik bleibt ein quantenmechanisches System, das sich im Grundzustand (Zustand minimaler Energie) eines zeitunabhängigen Systems befindet, auch bei Veränderungen des Systems im Grundzustand, wenn die Veränderung nur hinreichend langsam (also adiabatisch) passiert. Die Idee des adiabatischen Quantencomputers ist es, ein System zu konstruieren, das einen zu dieser Zeit noch unbekannten Grundzustand hat, der der Lösung eines bestimmten Problems entspricht, und ein anderes, dessen Grundzustand leicht experimentell zu präparieren ist. Anschließend wird das leicht zu präparierende System in das System überführt, an dessen Grundzustand man interessiert ist, und dessen Zustand dann gemessen. Wenn der Übergang langsam genug erfolgt ist, hat man so die Lösung des Problems."* *[Eintrag auf Wikipedia, aufgerufen am 18.1.2019]*

Der adiabatische Quantencomputer findet also das globale Minimum einer Funktion, die ein quantenphysikalisches System (d.h. das zu lösende Problem) beschreibt. Dieser Ansatz ist daher sehr gut geeignet um Optimierungsprobleme wie unser n-Damenproblem zu lösen.

## Chimera-Graph

Auf einem adiabatischen Quantencomputer sind nicht alle Qubits direkt miteinander verbunden (koppelbar). Der sog. Chimera-Graph beschreibt die Verbindung der physikalischen Qubits auf einem D-Wave Quanten Annealer. Er zeigt alle aktuell aktiven Qubits (Ausfälle einzelner Qubits sind möglich) und deren Verbindungen zu anderen Qubits. Universelle, vollständig miteinander verbundene Qubits werden daher durch das Zusammenfassen (Koppeln) einzelner physikalischer Qubits zu "logischen" Qubits realisiert (siehe Abbildung unten).

|  |  |
| --- | --- |
|  |  |

*Chimera Graph und Koppelung einzelner physikalische Qubits zu logischen, universellen Qubits (jede Farbe stellt einen logischen Qubit dar)*

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

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

*Beispiel für eine Hamiltonian-Matrix*

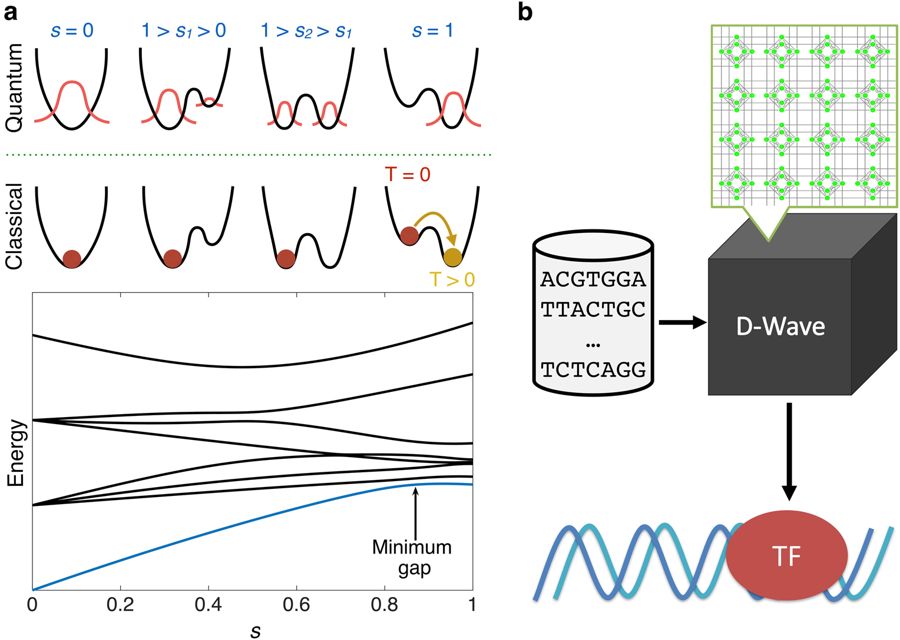
Der Hamiltonian ist die Basis für die Koppelungen der Qubits auf dem Chimera-Graphen. Er zeigt, in Form einer Matrix, welche Kombination zweier Qubits bestraft oder belohnt werden soll. Wenn man also nicht möchte, dass zwei nebeneinanderliegende Felder besetzt sind (z.B. auf den Feldern a und b) trägt 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 entspricht der vorher beschriebenen Energie.

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:

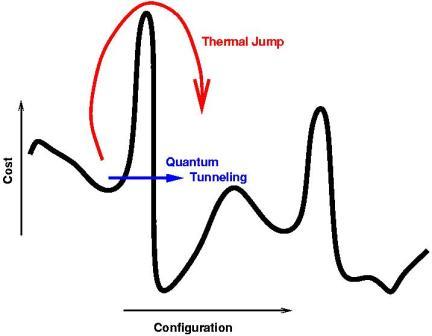
## Quantum annealing

Ein Durchlauf eines Quantum annealing Programms beginnt damit, dass, mittels eines Standard-Hamiltonians (initial-Hamiltonian), alle Qubits in Superposition „gesetzt“ werden. Langsam wird nun dieser Standard-Hamiltonian schrittweise mit dem eigentlichen Hamiltonian, also der Koppelung der Qubits auf dem Chimera-Graphen, ausgetauscht. Der Gesamt-Hamiltonian wird also wie folgt berechnet:



*Minimum gap in Zeit-Energie-Diagramm*

A wird im Laufe des Optimierungsvorgangs verringert, wohingegen B vergrößert wird. In der Grafik rechts sieht man (blau) den besten Energiezustand zu jedem Zeitpunkt des Annealing-Vorgangs. An der Stelle des „Minimum gap“ ist die Wahrscheinlichkeit am größten, dass man in einen schlechteren Energiezustand rutscht. Je nach Komplexität des Problems ist das Minimum gap größer oder kleiner.

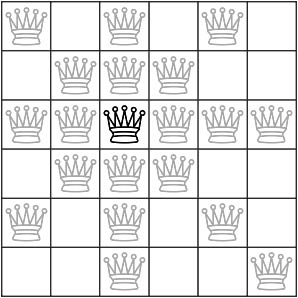


*Tunneling in einer Energielandschaft*

So ist es theoretisch möglich beim unendlich langsamen Übergang vom einen in den anderen Hamiltonian alle Erhebungen in der Energielandschaft zu tunneln (siehe Grafik rechts). In der Praxis reichen oft auch wenige Millisekunden, um das Problem zu lösen!

# Das n-Damen Problem

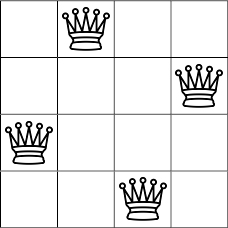
Das n-Damen Problem besteht darin, auf einem Schachbrett n Schachdamen so anzuordnen, dass keine Dame eine andere schlagen kann. Eine Dame darf immer vertikal, horizontal und in alle Richtungen diagonal ziehen. Erstmals formuliert wurde es 1848 von Max Bezzel, allerdings in Zusammenhang mit der Anzahl möglicher Lösungen für die Grundform n=8. Das Problem lässt sich sogar mit bruteforce (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 Optimiziation) und das Lösen auf einem adiabatischen Quantencomputer. Dieses Problem hat vor uns noch niemand auf einem adiabatischen Quantencomputer gelöst.



*Mögliche Bewegungsrichtungen einer Schachdame*

# Hamiltonian

Wir haben unseren Hamiltonian (siehe „Hamiltonian und Energieberechnung“) 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.



*Korrekte Lösung des 4 Damen Problems*

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.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 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 |

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.

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

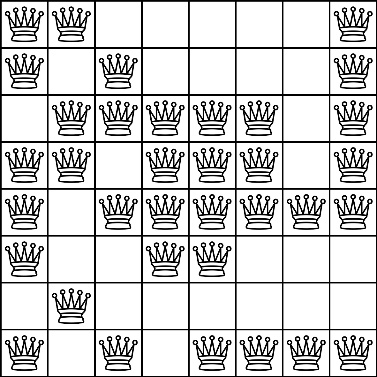
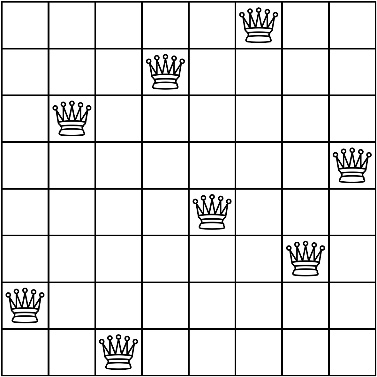
Vor dem Ausführen auf einem Quantencomputer haben wir unseren Hamiltonian noch mit mehreren Optimierungsverfahren getestet und diese verglichen. Diese Optimierungsverfahren wollen wir im Folgenden vorstellen. Alle Optimierungsverfahren haben ein richtiges Ergebnis geliefert.

## 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.

In der Folgenden Grafik sieht man die Energie in Relation zu der Anzahl an Durchläufen (entspricht der Anzahl der Änderungen).



## 

*Ausgewürfelte und endgültige Position der Damen beim greedy-Algorithmus*

## 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.

In der Grafik sieht man sehr gut, wie der Algorithmus am Anfang einen „Berg“ überläuft, um dann in einem globalen Minimum anzukommen.

## 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

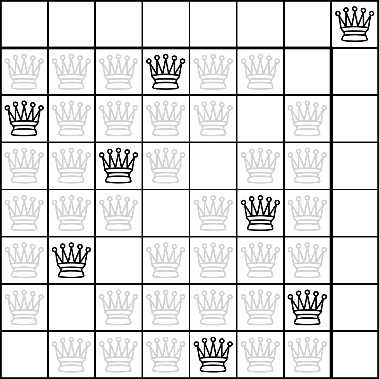
Der Great Deluge Algorithmus (englisch für Sintflut Algorithmus) lässt eine Änderung dann zu, wenn die neue Energie unter einem Schwellenwert liegt. Dieser Schwellenwert wird nach jeder Änderung herabgesetzt. Dieser Algorithmus hat mit unserem Hamiltonian sehr schlechte Ergebnisse erreicht. Um eine Lösung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

Unser Algorithmus hat, trotz vielem Testen, für den Hamiltonian schlechte Ergebnisse erzielt. Erst nach ca. 1,2 mio. Durchläufen konnte er ein Ergebnis finden.

# Auswertung der Ergebnisse des Quanten Annealers

Wir haben auf dem Quantencomputer „DW\_2000Q\_2\_1“ die Hamiltonians für alle n von 4 bis 8 ausgeführt. Als Eingabewerte bekommt der Quantencomputer, neben dem Chimera-Graphen, die Anzahl der Durchläufe und die Dauer eines Durchlaufs. Diese Dauer beträgt standardmäßig 20 Millisekunden, was sich auch bei uns bewährt hat. Im Folgenden Die Richtigen Ergebnisse pro 10.000 Durchläufe:

Da das 8-Damen Problem sogar bei 10000 Durchläufen kein Richtiges Ergebnis lieferte, gehen wir davon aus, dass es für den Quantencomputer zu schwer ist. Die Quantencomputer-technik ist noch ganz am Anfang und wird sich in den kommenden Jahren stark weiterentwickeln! Den Ansatz, welchen wir jetzt verwendet haben, ist sozusagen die „Vorlage“ für wirkliche große n, bei denen dann die Quanten-Überlegenheit eintritt, also der Quantencomputer bessere Ergebnisse als ein Supercomputer erzielt.



*Ergänzung des 7-Damen Problems zum 8-Damen Problem*

Um trotzdem ein Quantencomputer-gerechnetes Ergebnis für n=8 zu bekommen, haben wir uns überlegt, dass man, bei einer Lösung des 7-Damen Problems mit einer freien Gesamtdiagonale einfach eine Dame dazustellen könnte (siehe Grafik). Dafür haben wir unseren Hamiltonian umgeschrieben und konnten bei 10.000 Durchläufen auf dem Quantencomputer 80 mal ein richtiges Ergebnis für diesen Spezialfall finden!

Abschließend muss allerdings noch erwähnt werden, dass die richtigen Ergebnisse pro 10.000 Durchläufen sehr stark schwanken, da es nahezu willkürlich ist, ob der Quantencomputer im der besten Energiezustand bleibt. In den nächsten Jahren werden die Quantencomputer immer weiter verbessert, sodass diese Fehlerrate gesenkt werden kann. Dann wird man auch erst das ganze Potenzial ausnutzen können und die Quantencomputer werden in bestimmten Bereichen die herkömmlichen Computer bei weitem überholen!

# 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 Prof. Dr. Kristel Michielsen vom Forschungszentrum Jülich für die Rechenzeit am D-Wave Quantencomputer bedanken. Nicht zuletzt wollen wir Herrn Dennis Willsch, der beim Forschungszentrum Jülich im Bereich quantum computing arbeitet, für die vielen Erklärungen und Hilfestellungen rund um das Benutzen und Steuern des Quantencomputers danken.

Bis zum Präsentationstermin möchten wir noch versuchen, das „knights-tour problem“ (zu Deutsch Rösselsprung-Problem) auf einem Quantencomputer zu lösen. Hierbei soll ein Springer auf einem Schachfeld eine Route finden, auf der er jedes Feld genau einmal besucht. Wir hoffen, dass wir dieses weitere Schachproblem auf Basis unseres bisher erworbenen Wissens in kurzer Zeit ebenso lösen können. Einen vielversprechenden Ansatz für einen QUBO-Hamiltonian, der dieses Problem beschreibt, haben wir schon gefunden.

# Impressum

* D-wave Quantencomputer 2000Q: <https://www.theverge.com/circuitbreaker/2017/1/25/14390182/d-wave-q2000-quantum-computer-price-release-date>
* Darstellung des Spins eines Qubits: <https://www.nature.com/articles/nature10012/figures/1>
* Chimera-Graph (blau): <https://www.researchgate.net/figure/Chimera-the-graph-of-qubit-interactions-in-D-Waves-machine_fig1_263048587>
* Chimera-Graph Beispielkoppelung: <https://www.semanticscholar.org/paper/Fast-clique-minor-generation-in-Chimera-qubit-Boothby-King/39130a8ff551498da753acb0d9a3787f3f403b11/figure/1>
* Zeit-Energie Diagramm eines Anneals: <https://www.nature.com/articles/s41534-018-0060-8/figures/1>
* Energielandschaft (Tunneling): <https://en.wikipedia.org/wiki/Quantum_annealing#/media/File:Quant-annl.jpg>
* Alle nicht genannten Grafiken und Graphen sind selbst erstellt.