高效能巨量資料與人工智慧系統期末報告

組員：B05902108 施承志
B05611047 鍾宜樺

前言

如同老師在課堂中常常提到的，因為摩爾定律的衰落。並且在課堂中有小小的提到了量子電腦的部分與量子信息的進步。因此這次期末報告，我們研究了量子相關方面的研究與討論。

此份報告的內容大概可分成以下幾個部分：

第一部分、量子計算介紹

此部分會先探討量子計算、以及高效能計算如何在現階段量子計算不足的部分進行幫助。

第二部分、量子退火法

這部分會主要是對幾個常見的量子退火法做介紹，包刮：

(1) 退火法最起始的：Simulated Annealing (SA)、以及接續在量子上開始的量子退火法：Quantum Annealing (QA)。

(2) 加拿大的 D-wave、日本的 Hitachi 兩家公司在 Simulated Quantum Annealing (SQA)的研究。

(3) 日本的 NTT 與 Stanford 合作的 Coherent Ising Machine (CIM)。

(4) 日本的 Toshiba 所致力於的 Simulated bifurcation (SB)

等相關量子退火法的介紹。並在此部分中將會提及一些常見的量子退火解的問題，像是 Traveling salesman problem(TSP)、Max-cut problem、Number partitioning problem 等 NP-Hard 問題。

第三部分、實作實驗

於此部分中，我們針對前一個 part 所提到的部分實驗進，行更進一步做實驗與比較。其中包括：

(1) 東北大學把 Fully-connected Ising Problem Simulated Quantum Annealing 做在 FPGA 上的 work。

(2) Toshiba 在 Simulated bifurcation(SB)上面的 work。

最後由於報告篇幅的關係，將部分補充資料放置於 Appendix 與 Reference 中進行參照。

第一部分、量子計算介紹

量子計算始於 1980 年代早期，Paul Benioff 首先提出了運用量子力學的 Turing machine，之後 Richard Feynman 和 Yuri Manin 更進一步提出量子電腦可以做一些事情是傳統電腦無法有效率模擬的。

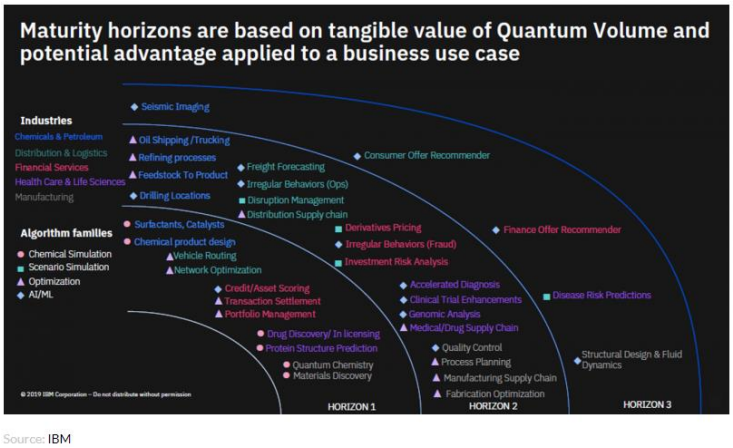
之後，第一個有用的量子演算法質因數分解於 1994 年 Peter Shor 提出，可以在多項式時間內做出質因數分解，相較於此，目前在傳統電腦最快的解質因數方法 sieve theorem 則是要 exponential time，儘管目前不知道質因數分解是否為 P，都可以輕易地了解到量子計算的強大。

時至今日，許多大公司都有他們其量子電腦的實現 如下圖 Fig.1 所示。量子電腦是可以透過這些大公司的雲端服務 access 的到的，不會是遙不可及的。

技術名稱	優點	缺點	代表公司
超導迴路	實踐快速、目前效果顯著	容易崩潰、必須一直保持低溫	Google, IBM, Intel, Microsoft
量子點	穩定、目前基礎	糾纏數量少，需長期保持低溫	IBM, Intel
囚禁離子	穩定，邏輯閘保真度 (fidelity) 高	運作緩慢、需要大型雷射系統	IonQ
拓樸量子位元	穩定度最高，能大幅降低錯誤率	理論最為困難，實作上相對不易	Microsoft, Bell Lab
鑽石空缺	可在室溫下操作	要達到糾纏態困難	Quantum Diamond Technologies Inc (QDTI)

Fig.1

量子計算在許多領域都有其可以幫到忙的地方，參考下圖 Fig.2，好比如：化學、醫療、金融…等。甚至是人工智慧方面，都能凸顯出了量子計算的重要性。



Source: IBM

Fig.2

然而，目前量子計算還沒有到成熟，可以被大量使用的地步，主要的原因是量子狀態的維持。量子狀態（其中包括：superposition、entanglement 等）持續存在的時間和所使用的技術有關係。如下圖 Fig.3 的 Longevity(量子態可

存在的時間)的 row 說明了這件事，可以看到被許多大公司常用的超導迴路 Longevity 非常短（超導迴路技術是建立在過往半導體技術，可以比較快上手)；相對來說 trapped ion longevity 很長，但是其每個計算所需要的時間也相對較長。整體來說，無論哪個技術，在其所能維持的量子狀態時間內，所能做的計算次數都是不多的。

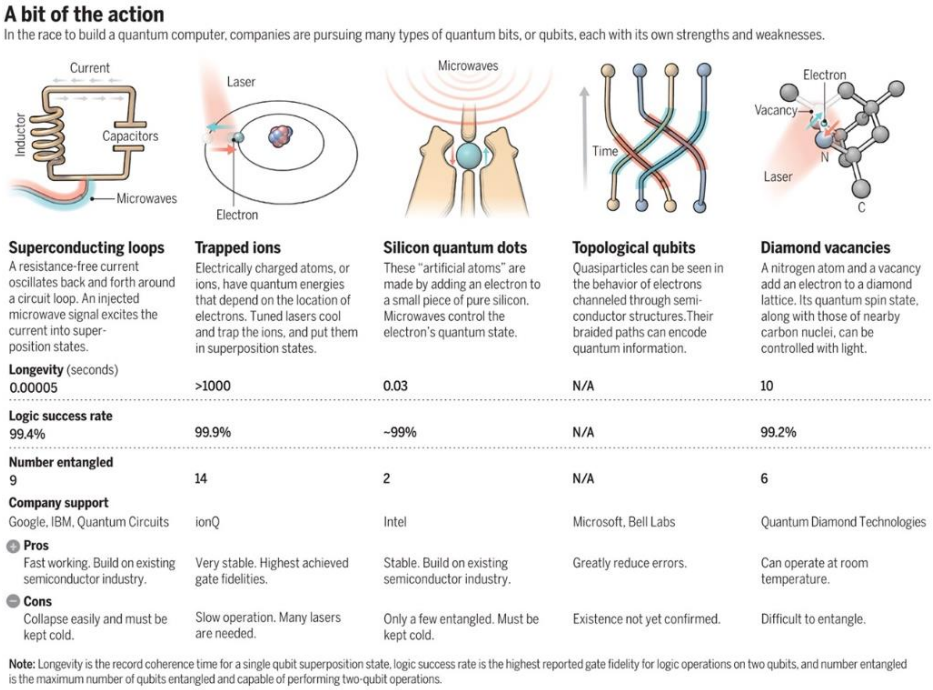


Fig. 3

雖然受限於這量子狀態時間的限制，但是人們也同時發展出了 quantum error correction。最一開始，同樣是由 Peter Shor 提出，主要是藉由 encode redundancy ，參照下圖 Fig. 4，以及曾時發生很多錯誤的機率是很低的，來做 error correction。然而，quantum error correction 所需要的 redundancy 是非常多的（一個 logical qubit 用百個 physical qubit)，在目前 IBM 53 qubits 是還有一段路要走的。但是一旦這 correction 技術成熟後，之後 scale up 會非常快，達到真正的量子計算。

Triplet received	Interpreted as
000	0 (error free)
001	0
010	0
100	0
111	1 (error free)
110	1
101	1
011	1

Fig. 4

在目前階段所能同時使用的 logical qubit 是只有個位數個的，這也是現在常看的一些雲端提供出來的 quantum circuit 往往只有幾個可以使用 ，如下

圖 Fig.5 所示。量子計算現在處於一個於 2018 提出來的名詞 NISQ (noisy intermediate-scale quantum)的階段，所用的 qubit 和其對應的 operation 都是非常 noisy 的，且 scale 還不是很大。

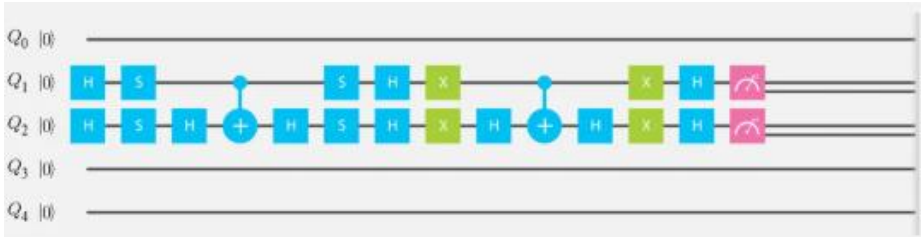


Fig.5

使用傳統電腦或超級電腦來模擬量子電腦的 work 也是不在少數，如 Fig.6。主要是當量子系統還可以被模擬的狀態下，我們可以檢驗其正確性，以及可以知道量子電腦的哪個環節出錯了，去幫助到量子電腦的發展。等到我們無法繼續模擬時，也許未來會想出另一套方法來做驗證。

Reference	General Technique	Qubits	Depth	# of Amplitudes
Intel [6]	Full amplitude-vector update	42	High	All
ETH [5]	Optimized full amplitude-vector update	5 × 9	25	All
IBM [7]	Tensor-slicing with minimized communication	7 × 7 7 × 8	27 23	All 2^{37} out of 2^{56}
Google [8]	Preprocessing using undirected graphical model	7 × 8	30	1
USTC [9]	Qubit partition with partial vector update	8 × 9	22	1
Sunway [10]	Dynamic programming qubit partition	7 × 7 7 × 7	39 55	All 1
Alibaba	Undirected graphical model with parallelization	9 × 9	40	1

Fig.6

其實 google 於 2019 年 10 月提出了 Quantum supremacy 的概念，其實就是設計一套 Highly-entangled quantum circuit (N-fully entangled qubits 需要2^{n+1} 個 Floating points 來模擬的，是成指數成長的)。在他們的 quantum computers 上能夠跑得起來。但是，即使是用目前最好的超級電腦仍需要要花不少時間去做模擬。

儘管如此，現今傳統模擬還是可以幫助量子電腦的發展，畢竟量子電腦不是人人都可以無上限的使用。另外，構建量子計算的理論已經存在許久並獲得了許多數學物理方面的驗證。並且我們可以做到模擬它，並做一些相對的實驗。雖然目前的 scale 上較少，但是或多或少可以幫助到使用量子計算的社群。

第二部分、量子退火法

前面簡短的介紹量子計算目前的現況，接下來會 focus on 量子計算其中的一支，量子退火法。量子計算可分為 Gate based system 和 Ising machine system。

前者是很多大公司做的，gate based 可以做到比較 universal 的 operation，可以找規律很快、找 eigenvalue 很快以及解方程式很快，後者可以做到的事情 gate based 也可以辦得到。

至於後者 Ising machine system，也就是利用量子退火的方法來做的(Quantum Annealing)，主要是 focus on optimization problem，有真正用到量子物理的是一家加拿大的 D-wave 公司在做的，下圖可以看到還有其他不是使用真正的量子物理而是用傳統電腦模擬，或是利用其他系統如光學來模擬量子現象的公司，大都是日本公司。

接下來會 focus 在這些利用模擬的公司，來進一步帶出我們相關的 work。

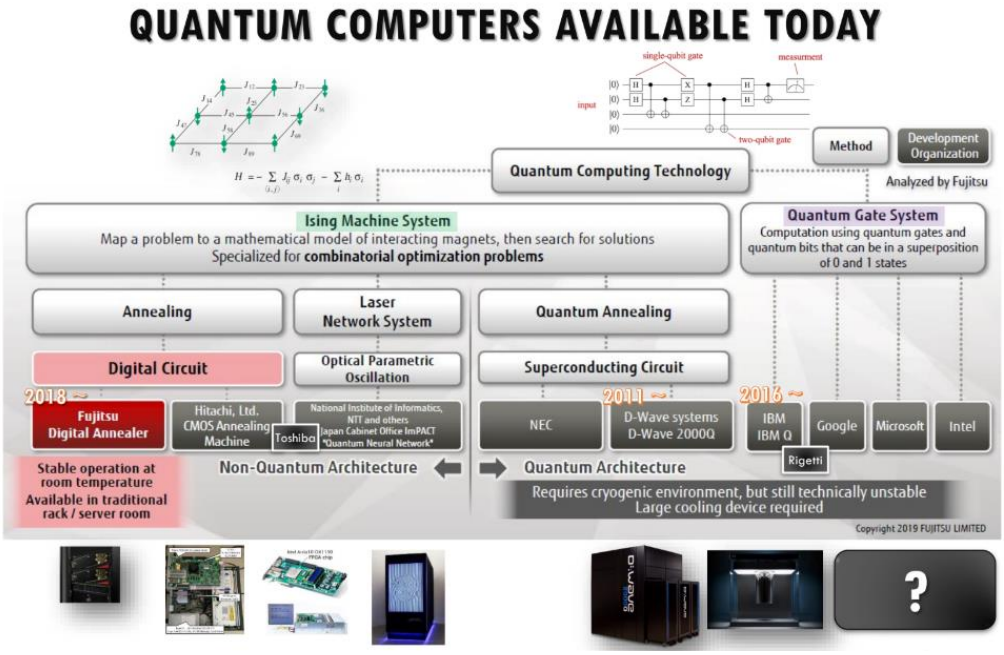


Fig. 7

退火法其實在 1953 年就有提出了，其一開始是物理相態的模擬，藉由慢慢降溫來讓系統溫度達到最低，這中間可以觀察到相態的變化。後來可以拿來把 optimization problem encode 進去系統，當溫度達到最低時，剛好就是 solution 的解，常被拿來解的問題包括 Traveling salesman problem、Max-cut problem、Number partitioning problem…等 NP-Hard 問題。

Simulated Annealing (SA)

退火法運作的方式如下圖 Fig.8，我們舉 Traveling salesman 為例，我們會定義一個 cost function，在每一種可能的答案都有其對應的值，這裡我們定義的 cost function 就是我們走的路徑長，而我們要 minimize 這 cost function，所以可以想像當 cost function 值最低時，就應該要是答案了。再來我們會決定從一個可能的答案(state)是否要 transition 到另一個可能的答案(state)，criteria 是下圖的 12 行，我們會把溫度 T 拿進來一起討論，這個溫度會從高到低慢慢降溫。

如果$\Delta E<0$（兩個不同 state 的 cost function 相減），也就是代表要轉移過去的 state 的溫度比較低，這時$e^{-\Delta E/T}>1$，一定會大於我們擲的骰子 rand(0.1)而因此 transition 過去。

如果$\Delta E>0$，也就代表要轉移過去的 state 的溫度比較高，這時$0<e^{-\Delta E/T}<1$，有一定機率轉移過去，但也和溫度有關，當溫度一開始高時，$e^{-\Delta E/T}$會比較接近 1，而比較容易轉移過去，隨著溫度慢慢降低，轉移過去的機率下降，因為這時系統其實已經差不多要達到最低溫度，不太傾向往能量高的 state 轉移過去。

可以看到藉由這個方法，且我們做的夠久，最終一定能達到最低溫度而剛好得到最佳解，加入了在一開始可以從低溫度 state 跳到高溫度 state，使我們可以跳出 local optimum，而最後到達 global optimum。

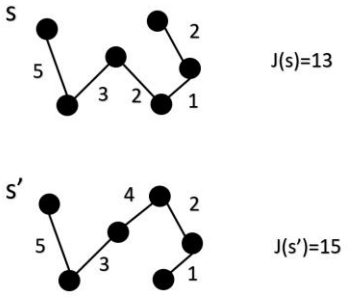


Fig.8

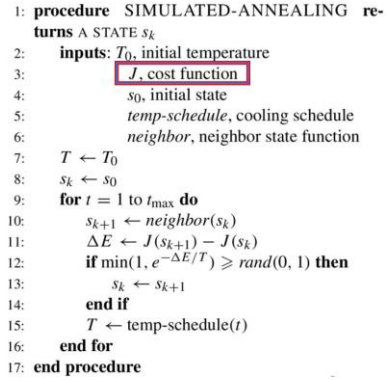

1: **procedure** SIMULATED-ANNEALING **returns** A STATE s_k
2: **inputs**: T_0, initial temperature
3: J, cost function
4: s_0, initial state
5: *temp-schedule*, cooling schedule
6: *neighbor*, neighbor state function
7: $T \leftarrow T_0$
8: $s_k \leftarrow s_0$
9: **for** $t = 1$ to $t_{\max}$ **do**
10: $s_{k+1} \leftarrow neighbor(s_k)$
11: $\Delta E \leftarrow J(s_{k+1}) - J(s_k)$
12: **if** $\min(1, e^{-\Delta E/T}) \geqslant rand(0, 1)$ **then**
13: $s_k \leftarrow s_{k+1}$
14: **end if**
15: $T \leftarrow$ temp-schedule(t)
16: **end for**
17: **end procedure**

Fig.9

上面講了退火法簡單是如何進行的，回到大部分日本大公司做的退火法，只是把他們的 cost function 換成 Ising model，參考下圖 Fig.10：

(1) σ的值是+1、-1 代表我們要的答案、(2) J 代表σ之間的 interaction term、(3) h 代表σ自己的 term。

幾乎所有的 NP-hard problem 都可以 encode 成這種形式，這篇給出了不少 formulation，藉由讓 Ising model 能量達到最低，我們讀出最低能量的σ，這+1、-1 就通常代表了答案，如 Traveling salesman 某個城市走或不走。

Ising model: $H(\sigma) = -\sum_{(i,j)} J_{ij}\sigma_i\sigma_j - \sum_i h_i\sigma_i$

Fig.10

Quantum Annealing (QA) 、Simulated Quantum Annealing(SQA)

除了前面提到在 1953 年就有的一般的傳統退火法，另一個東西叫做量子退火法 Quantum annealing，參考下圖 Fig.11。可以看到右項和傳統 Ising model 幾乎是一樣，只是多了左項。在一開始 A(s)比較大，B(s)是 0，所以一開始只有左項，之後 A(s)慢慢變小，B(s)慢慢變大，右邊我們要的那一項慢慢出來，最後得到我們要的答案。這邊用到了一個叫 Adiabatic theorem 的物理性質。簡易物理性質介紹是：如果我們一開始能讓左邊那項能量最低，然後慢慢把右邊那項 inject 進來，只要做的夠慢，右邊那項最後出來時能量也會是最低的。這是 D-wave 那家公司在做的，是使用了真正的量子力學。

之後再講到這個量子退火法是可以被模擬的，第一部分有提到要模擬一個量子系統，我們需要2^{n+1} 的 exponential 的空間放在 memory 去做模擬。除了用 exponential 的空間去做模擬，有別的近似的模擬方法使用了 Path Integral（於 SQA 所使用的方法）的方法來做的，只需要 linear 的空間就可以做模擬，參考下圖 Fig.12、Fig.13。

$$\mathcal{H}_{ising} = \underbrace{-\frac{A(s)}{2}\left(\sum_i \hat{\sigma}_x^{(i)}\right)}_{\text{Initial Hamiltonian}} + \underbrace{\frac{B(s)}{2}\left(\sum_i h_i\hat{\sigma}_z^{(i)} + \sum_{i>j} J_{i,j}\hat{\sigma}_z^{(i)}\hat{\sigma}_z^{(j)}\right)}_{\text{Final Hamiltonian}} \qquad \mathcal{H}_{d+1}(t) = -\sum_k^P \left(\sum_{i,j}^N J_{ij} z_i^k z_j^k + \sum_i^N h_i z_i^k + J_\perp \sum_i^N z_i^k z_i^{k+1}\right)$$

Fig.11、Fig.12

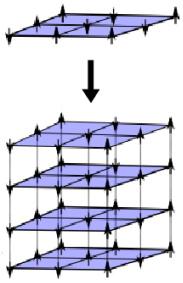


Fig.13

上面我們簡約 cover 了 SA、QA、SQA。利用傳統電腦模擬的有 SA,SQA，這是日本大公司有在做的，包括 Fujitsu, Hitachi。除了有用傳統電腦模擬，他們還有用 FPGA, GPU 甚至是 ASIC 來做這些事情；利用真正的量子物理在做的有 QA，這是加拿大公司 D-wave 在做的。下圖列了一些相關的圖片：

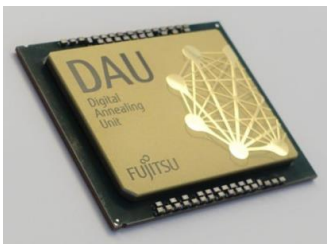


Fig.14 - Fujitsu Digital Annealer、

Fig.15 - Hitachi CMOS Annealing machine、Fig.16 - D-wave quantum computer

Coherent Ising Machine (CIM)

除了用傳統電腦、FPGA、GPU 和 ASIC 來模擬退火，接下來要介紹運用光學來模擬退火方法，Coherent Ising Machine (CIM)。為了達到高效能的計算，量子電腦與量子計算與其他非傳統計算方案開始受到的重視並且被大量研究，其中 CIM 便是其中一種。CIM 是從 2003 年日本的電信公司 NTT 與 Stanford 的 Peter L. McMahon 教授合作的一項企劃。並且在 2016 年加入了日本東京大學的量子光學大師，山本喜久。

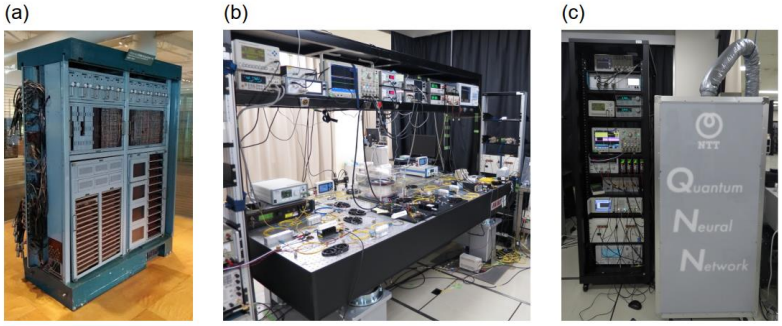


Fig. 1. (Color online) Old and new DPO-based computers. (a) MUSASINO-1B, the parametron computer developed by Nippon Telegraph and Telephone Public Corporation in 1960 (photo by courtesy of NTT History Center of Technologies). (b) The first CIM experimental setup constructed on an optical table. (c) The CIM system in a soundproof box with temperature control used for the cloud Max-Cut computation service.

Fig.17

CIM 的機器演進大略如上圖 Fig.17 所示。其中，最右圖是現今用於解決 MAX-XUT 的 CIM 的實體機器的外觀。已知，CIM 是用於解決 Ising problem 的 Ising machine，而 CIM 所使用的 Ising spin 則是 Degenerate optical parametric oscillator，在不同的 paper 中有時被稱為 OPO、亦可稱為 DOPO。

先對 CIM 的概念做介紹，CIM 是利用 laser 打出 photon pump，並在這些雷射光波經過不同的光學儀器、非對稱性的化學結晶，使得光產生了二次諧振態的光波作為 Ising machine 中的 Ising spin，也就是我們的 OPO。在產生 OPO 之後便開始做 coupling 的動作，並做降能，類似於做 flipping spin 的動作，也就是改變 OPO 的 phase and amplitude 的 state，盡可能在所設定的 annealing time 去找到最低能量(Hamiltonian)的 configuration，即找到所求，或者是即使不是索求也盡可能為相對最佳解。

其中影響 performance 的因素包刮：size，N 的大小、annealing time、問題是否為 fully connected、Degree 的大小…等等。

CIM 的物理機器介紹

CIM 的物理機器可以分成兩的部分。第一部分是 OPO Part，也就是機器產生 OPO (Ising spin)的部分；另外是 Feedback Part，也就是機器決定是否要做 coupling 的部分。其中整個 OPO part 的過程都是在一個腔體中進行，而被產生的 OPO 會在光纖所形成的管路中一直跑，直到整個系統的 Hamiltonian 降到最低，也就類似於退火的過程結束。

除此之外，CIM 的物理機器大略可分成三代機器，分別是在 2003-2012 的

第一代，團隊在實驗桌上進行小型的 CIM 實驗。2014 是 Stanford 與 NTT 合作的 CIM 的實體機的出現。以及 2016 後使用 FPGA 作為決定是否要做 coupling 的主要算力的第三代機器。其中第三代機器也從 2016 一路使用到 2020 在硬體結構上並沒有太大的變化。

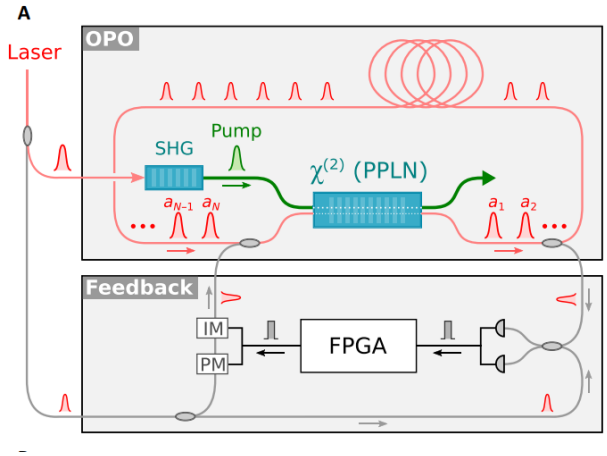


Fig. 18

首先放上 CIM 的 full machine diagram 如 Fig. 18，這是 CIM 第三代的機器的架構圖。但因為從第一代到第三代的 OPO part 基本上大同小異，故先使用第三代的架構圖講解 OPO 是如何產生的。一開始，會由左上角的 laser 不斷的打 pump。當 pump 在被打進腔體後，pump 會先經過兩種在物理光學上常常被用到的光學機器以及非中心對稱的結晶體，通常會為一組 Parametric amplification 一起被應用，如下圖 Fig. 19 所示。Second Harmonic generation，簡稱 SHG；Periodically poled lithium niobate，簡稱 PPLN。

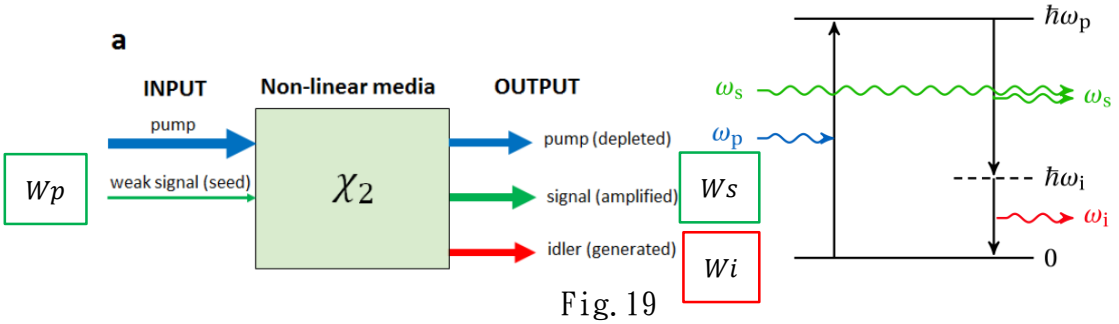


Fig. 19

被打出的 pump 會先經過 SHG，次諧波生成（又稱倍頻）是一種非線性光學過程，如圖 Fig. 19 所示。其中具有相同頻率的兩個光子與非線性材料相互作用，被“組合”，並產生具有初始光子能量兩倍的新光子（等效地，兩倍的頻率和一半的波長），可以保持激發的相干性。這是和頻生成（2 個光子）的一種特殊情況，更常見的是諧波生成。簡單來說，就是將原本頻率為Wp的 pump，在經過 SHG 後，會分裂成兩個光波，分別稱為 signal 以及 idler，且因為能量守恆，因此可知$Wp = Ws + Wi$。當 $\frac{1}{2}$ $Wp = Ws = Wi$，這種特殊情況被稱為 Parametric down-conversion(PDC)。

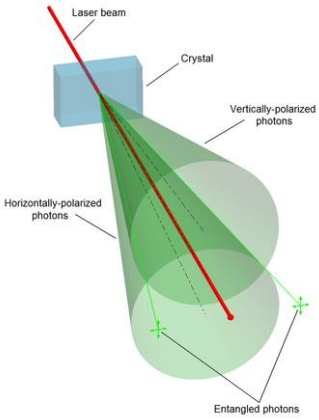


Fig. 20

CIM 便是採用使得$\frac{1}{2}$ $Wp = Ws = Wi$，也就是 PDC。而在常用的 PDC 設備設計中，強激光束對準 BBO（硼酸鋇)晶體。大多數光子直接穿過晶體，如圖 Fig. 20。但是，偶爾某些光子會進行 II 型極化相關的自發下變換，並且所得的相關光子對的軌跡沿兩個圓錐的邊緣受到約束，其軸相對於泵浦光束對稱排列。同樣，由於動量守恆兩個光子始終相對於泵浦光束沿圓錐體的邊緣對稱放置。重要的是，光子對的軌跡可能同時存在於錐相交的兩條線中。這導致偏振垂直的光子對糾纏。簡單來說，此光波會在經過 PPLN。PPLN 的作用是為了達到 entanglement 的作用，讓產生出來的 OPO 能因為出現機率的不同，而有更多的變化性。其中 CIM 所使用的非線性對稱的結晶是硼酸鋇。另外 PPLN(硼酸鋇)也有另一種功用是作為 phase-sensitive amplifier(PSA)，他只能讓 pump-phase$\in\{0, \pi\}$的光波，也就是我們所需要的 Ising spin 通過。而在經過 SHG 和 PPLN 後的 pump，就是我們所需要的 OPO，Ising spin。

提完 OPO 產生的部分要討論在 Ising problem 中，Ising spin 在 CIM 是怎麼做 coupling 的部分。而在這部分物理機器則分成三代。

First generation 架構圖如 Fig. 21 所示，fs laser 會和第三代機器一樣，不斷打出 pump laser，並使用 clock（CP）控制當下打出的光波和前一個打出的光波中的間隔時間，使原本連續的光變成一個接著一個的 pump。而在光波被打出後會因為 M1~M4 這四面鏡子的折射與反射，讓光波在圖中以虛線框框起的路徑中行徑。其中，在路徑中有置放有一個 PPLN 而就如同上一段所提到的一樣，在製造出所需要的 OPO 之後便會開始進行 couplings 的部分。第一代機器做 couplings 的方式相對簡單，是利用 delay gate 的方式進行。舉例來說，假若希望讓 OPO1 對 OPO2 做 coupling，則在 OPO1 那一輪的 circulation 時，在 delay gate-1 上的 OC 與 IC 兩種分光儀，會將一部份的 OPO1 的光波分到 delay gate-1 裡面，並讓分進去的光在這個管路中行徑並反射最後再回來。剩下的 OPO1 則繼續行徑。其中被分進 delay gate-1 裡面的光，則會在下一輪的 OPO2 的 circulation 行逕至 delay gate-1 的 IC 時候，OPO2 與從 delay

gate-1 中跑出來的先前被分出去的 OPO1 合併成一個新的 OPO2，即完成了希望讓 OPO1 與 OPO2 做 coupling 的動作。同理，若希望讓 OPO2 與 OPO4 做 coupling，則在 OPO2 的 circulation 中，會分一部份的 OPO2 進 delay gate-3，並再 OPO4 的 circulation 的時候，被分出的 OPO2 會與 OPO4 會合進行 coupling。如同 Fig.4 所示，希望 A 跟 B 做 coupling 時，應該將哪一個 OPO 放進 delay-gate，與哪個 delay-gate 裡面。最後再 first generation 要特別提到的是，在初代機器只是測試 CIM 的方法是否可行，且另外因為光速行徑的時間非常快，故初代機器是非常小型，且是在實驗桌上進行的。

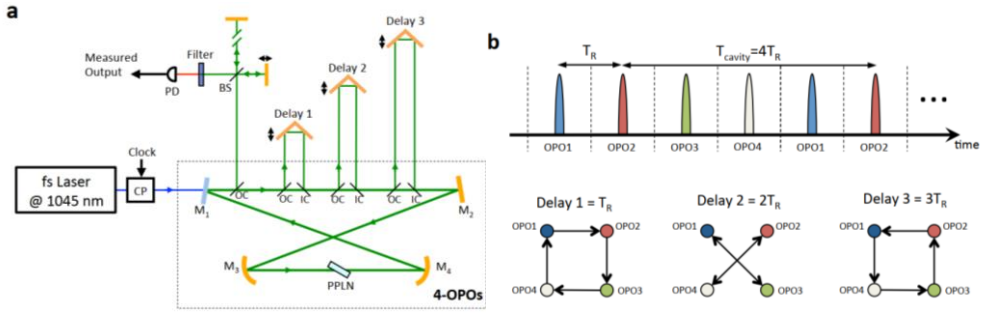


Fig.21 First generation CIM machine, the way doing coupling

Second generation，在確定 CIM 的概念可行後，團隊便開始實作真正的實體機器，如 Fig.22 所示。基本上在產生 OPO 的部分是相同的，不同的點是做 coupling 的部分。Coupling 概念相同的，不同的是原本的 delay-gate 會全部改成 EOM，電光調製器。用於可以用來調製強加相位，頻率，振幅，或偏振的光束。通過使用激光控制的調製器，可以將調製帶寬擴展到千兆赫茲範圍。光在經過 splitter 的後，會選擇進入不同的 EOM，也就是類似於不同的 delay-gate，在 EOM 中會保存要被拿去做 coupling 的光波，其中 EOM 也會補光、調製 OPO 在 EOM 中的 phase，要做這些事情的原因是因為光不能存在在光纖中太久會消散。而在要做 coupling 後會在經過 combiner 傳送進去光纖裡面做 coupling。

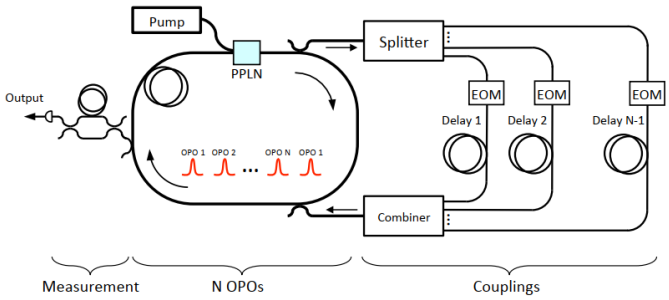


Fig.22 Second generation CIM machine

然而，因為 EOM 非常貴，因此團隊又做了機器上的更新，因此有了第三代的機器。Third generation，如下 Fig.23 所示。產生 OPO 的地方依舊不變。改變的依舊是 coupling 的部分。不同於前面會是一個 OPO 與另一個 OPO 做

coupling，在第三代的機器時，在每一次的 circulation 中，會是一個 OPO 決定要不要和除了自己以外的全部的分別做 coupling。也就是說，假設全部有 4 個 OPO，則在 OPO1 的 circulation time 中，可以同時決定是不是要跟 OPO2、OPO3、OPO4 做 coupling。

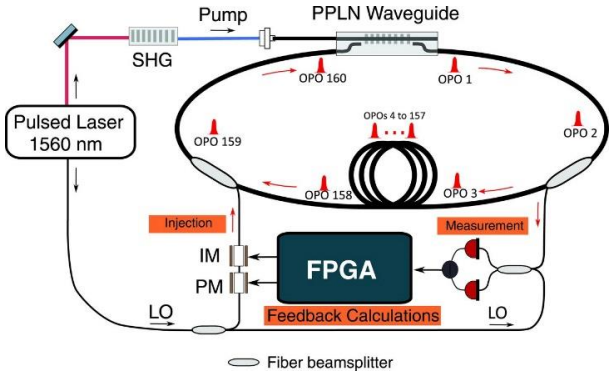


Fig.23 Third generation CIM machine

$$G = \begin{pmatrix} 0 & -\xi_{12} & \dots & -\xi_{1N} \\ -\xi_{21} & 0 & \dots & -\xi_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ -\xi_{N1} & -\xi_{N2} & \dots & 0 \end{pmatrix}$$

Fig.24 Matrix store in FPGA

不使用 EOM 後，CIM 的團隊改採用 Measurement and feedback，MFB 的方式。假設現在是 OPO2 的 circulation time，則 OPO2 如圖右中的部分，會先進去灰色的長橢圓形，也就是分光儀。OPO2 會被分出一些光進入下面做 Measurement，其中被測量參數的包括 phase 跟 amplitude，而這測量的部分同時也是 Analog-to-digital converter，會將這些測得的光波的參數值傳進 FPGA 中。由 FPGA 去做 calculation 決定如何做 feedback 的部分。而 FPGA 中存的是 scaling factor 的 matrix，如 Fig.24 所示。為什麼會存$\hat{\xi}$的原因，在之後敘述 Hamiltonian 的時候會提到$\frac{\hat{\xi}}{2}$代表的是表示 the j-th OPO 和 l-th OPO 之間 coupling 的強度。FPGA 會計算 feedback signal: $f_i = -r\sum_j J_{ij}\bar{c}\iota$（r 是常數），並根據回饋的訊號決定做 coupling 與否，讓整體的能量越來越低，若做 coupling 會使能量變低，則做；反之，不做。

那 CIM 為什麼可以使用 FPGA 計算是否做 Coupling 的原因是因為，CIM 的整個系統基本上左上角的 laser pump 會一直打，也一直持續增大 pump rate 的大小、將其打高，隨著 pump rate 越來越高，會因為 pump rate 和 OPO 的 amplitude 的實部，也就是 in-phase component 成正比，當 pump rate 越來越大的同時，in-phase component 也會越來越大，實驗結果如同下圖 Fig.25 所示。當能夠被測得的光波震幅越來越大後，如同 Fig.26 和 Fig.27 所示，Ising Energy 同時也在降低。並且在 pump rate 達到 groud state 時會產程 bifurcatiton，，並且很快就會得到我們所求，能量最低的狀態。

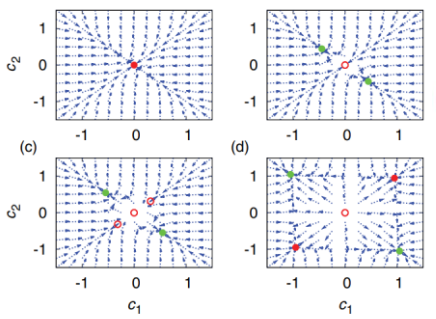


FIG. 2. (Color online) Phase diagrams for the in-phase components of two coupled degenerate OPOs when $\xi = -0.1$. The normalized pump rate for each diagram is (a) $p = p_{th} = 0.9$, (b) $p = 1.1$, (c) $p = 1.2$, and (d) $p = 2.0$. The dots and circles mean stable and unstable steady states, while the colors green (light gray) and red (medium gray) denote correct and incorrect solutions, respectively.

Fig. 25

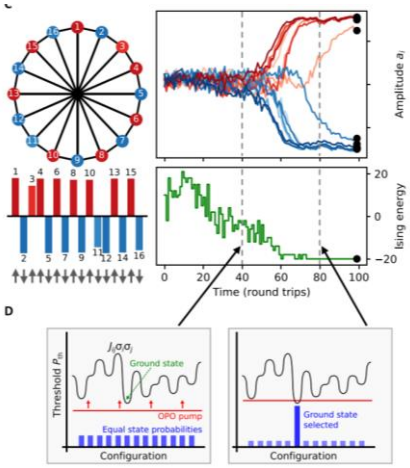


Fig. 26

最後再補充一點是，為什麼 CIM 的光波(OPO)能夠一直在光纖中存在而不消散，原因是因為如果他的機器架構圖中顯示的，在 Fig.1 的左上角的 laser pump 下面有接出另一條線路，這條線路便是用來補充 pump energy 進入各個 OPO 中，使 OPO 在很長的光纖中不消散的原因。

另外，我們有對 CIM 的 Hamiltonian 與作用理論進行相關的探討、以及 CIM simulation on classical computer。在 2018 年的時候，D-wave 在 arxiv 上發了一篇 CIM 的 simulation 的論文。以 CIM 是將 Ising problem 的能量最小化， D-wave 發表的方式則是利用與 CIM 類似的 noise，進行比較平均場退火的演算法。而此二部分，由於報告篇幅的關係，將放置於 Appendix 中。

Simulated Bifurcation (SB)

最後講到類似 CIM 的做法，是 Toshiba 的 work，一樣是在解 Ising model 最低的能量。藉由 encode Ising model 在一個系統內的每個彈簧的位置和動量(如下圖，x 是位置，y 是動量，$\dot{x}$的點是對時間微分也就是速度，$\dot{y}$是加速度，藉由不斷去更新位置和動量，讓系統趨向最低能量)，下去模擬這個系統，最後藉由觀察每個彈簧的位置的正負號來得到 Ising problem 的最低能量解。

$$H_{\mathrm{SB}}(\mathbf{x},\mathbf{y},t) = \sum_{i=1}^{N}\frac{\Delta}{2}y_i^2 + V(\mathbf{x},t) = \sum_{i=1}^{N}\frac{\Delta}{2}y_i^2 +$$

$$\sum_{i=1}^{N}\left[\frac{K}{4}x_i^4 + \frac{\Delta - p(t)}{2}x_i^2\right] - \frac{\xi_0}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}J_{i,j}x_i x_j$$

$$\dot{x}_i = \frac{\partial H_{\mathrm{SB}}}{\partial y_i} = \Delta y_i$$

$$\dot{y}_i = -\frac{\partial H_{\mathrm{SB}}}{\partial x_i} = -\left[Kx_i^2 - p(t) + \Delta\right]x_i + \xi_0\sum_{j=1}^{N}J_{i,j}x_j$$

Fig. 27

上述方法 Toshiba 稱作為 SB (simulated bifurcation)，主要是因為演化的過程中位置到某個時間點會分岔(bifurcation)，同時是正也是負(superposition)。運用這個方法一樣可以得到 Ising model 不錯的解。

第三部分、實作實驗

前面講了不少解 Ising model 的方法，接下來這個部分會帶出他們方法得到的結果，以及我們做的相對應的實驗。

Ising model: $H(\sigma) = -\sum_{(i,j)} J_{ij}\sigma_i\sigma_j - \sum_i h_i\sigma_i$

一開始講到 Ising problem 時，我們會先做分類，或者說有很多種方法可以解 Ising problem，就像前面提到的 SA, SQA, SB, CIM，除此之外，還可以做前處理如 graph partition, minor-embedding 等 technique。所以大致的 pipeline 是如下圖 Fig.28 所示。

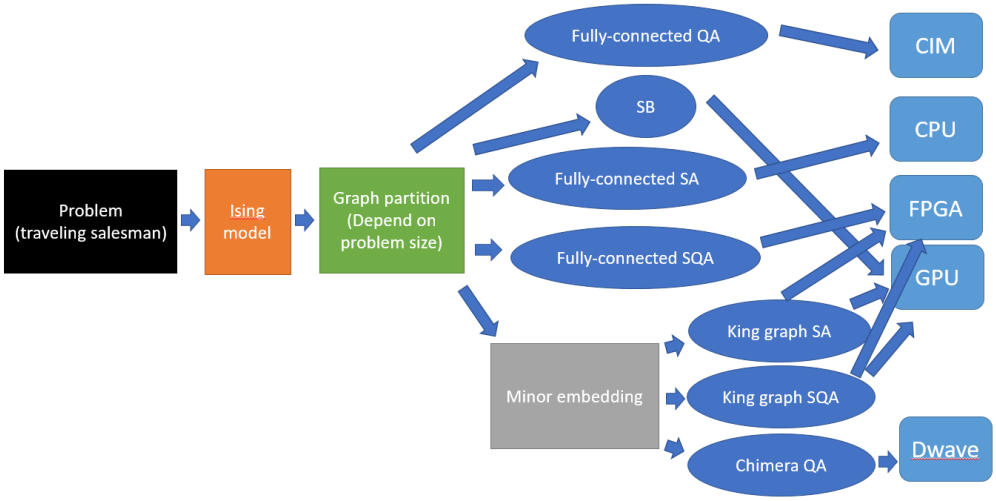


Fig.28

Graph partition 的理念是將一個大的 Ising problem 分很多小 part，去解 Ising problem，最後再 combine 起來 ，參考下圖 Fig.29，如 D-wave 有在做的 Tabu search algorithm。

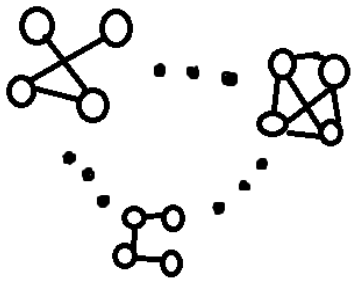


Fig.29

Minor-embedding 則是將 graph 塞進計算架構上去做計算，例如下圖 Fig.30 把 fully-connected ising problem 塞進 2x2 的 checkerboard 裡面。

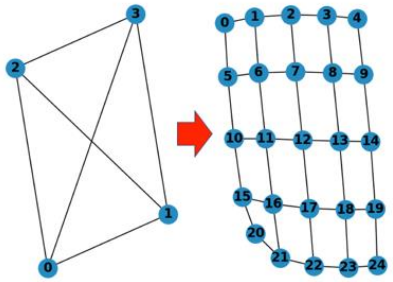
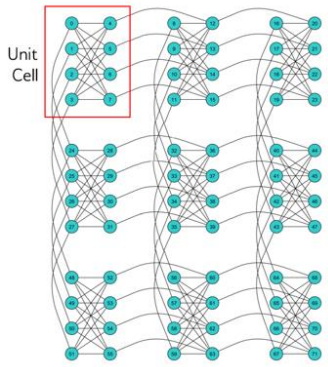


Fig. 30

Fully-connected Ising model 物理裝置是做不出來的，是 scalable 的。而每個公司所提供的計算架構不盡相同，如下圖，D-wave 所用的是 chimera graph，Fig. 31，而 Hitachi 在 SQA 用的是 king graph，Fig. 32。



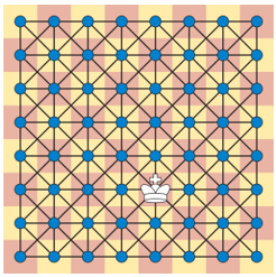


Fig. 31　　Fig. 32

Embedding 到 target device 後，我們可以用我們想要用的退火法，如 SA, SQA, SB, QA 等方法下去做退火，這些退火法可以在各種 device 上面跑，各有優缺點。

接下來會 focus on 一些 work 在 FPGA 和 GPU 的退火法。

首先是東北大學把 fully-connected Ising problem SQA 做在 FPGA 上的 work，由於是 fully-connected 同一層的 spin 不能同時 update，但是一旦第一層的某個 spin 好了，對應第二層的那個 spin 就可以開始 update 了，是利用 pipeline 的方式 ，參考下圖 Fig. 33。

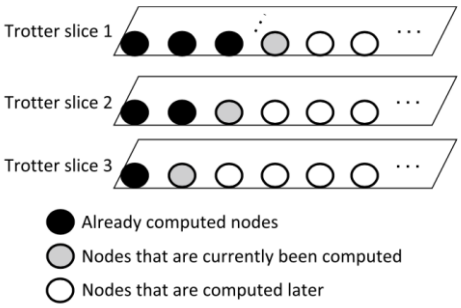


Fig. 33

他們所使用的 FPGA，Nallatech 385A accelerator boards，Intel Arria

10 10AX115N3F40E2SG，用的語言是 OpenCL：

FPGA
- Intel Arria 10 GX
  - 1150 GX F1517 NF40 package
  - Core speed grade –2: I/O speed grade –3
- Contact BittWare for other Arria 10 GX options

On-board Flash
- Flash memory for booting FPGA

On-board memory
- Two banks of DDR3 SDRAM x 72 bits
- 4GB per bank (8GB total /16GB and 32GB version also available)
- 2133MT/s per bank

Host interface
- x8 Gen3 interface direct to FPGA

QSFP cages
- 2 QSFP+ cages on front panel connected directly to FPGA via 8 transceivers
- User programmable low jitter clocking supporting 10/40 GbE
- Each QSFP can be independently clocked
- Clocking options:
  - Network recovered with jitter attenuation
  - QSFP clocking: user programmable, or CPRI, 1GbE
  - External clock input, 1PPS input

我們用的 FPGA 規格，Intel® Programmable Acceleration Card (PAC) with Intel® Arria® 10 GX FPGA：

FPGA	Intel® Arria® 10 GX FPGA
邏輯元素 (LE)	1150000
晶載記憶體	65.7 Mb
DSP 區塊	3036
Memory Specifications	
外部板載 DDR4	8 GB (4 GB x 2 banks)
I/O Specifications	
PCI Express 修訂版	3
PCI Express 設定 ‡	Gen3 x8 (electrical), Gen3 x16 (mechanical)
QSFP 介面	x1
USB 配置	USB 2.0
網路介面	10 Gbps, 40 Gbps (up to 40 GbE)

規格幾乎是差不多，synthesizer 也差不多都是 Intel FPGA SDK for OpenCL 17.1，除了我們這片是 DDR4 他們用的是 DDR3。

下圖 Fig.35 是 compile 出來的 system diagram，可以看到藉由 unroll 大量的 loop，很多計算可以在幾乎一個 cycle 裡面做完，只是所需要的資源較多。

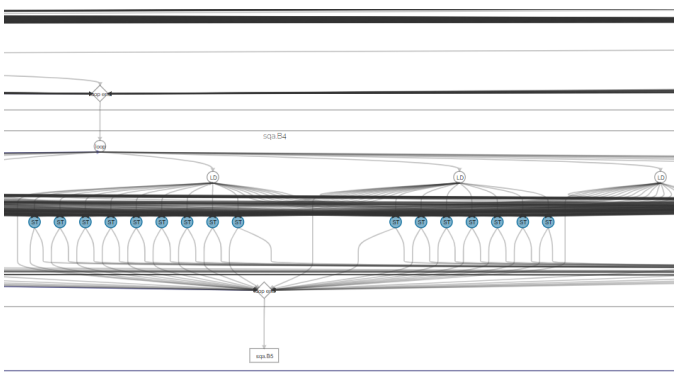


Fig.35

下面幾張圖是他們的實驗結果，可以看到 node size 上升所花的時間應該要和乘上 4 倍，但因為用了更多 resource，FPGA operate 在較低的 frequency，所以會再慢一些。而當 trotter size 上升時，FPGA 就不會像 CPU 一樣要 double 時間，因為有 pipeline 過了。

Table 1 Processing time versus node size (500 iterations)

Trotter size	Node size	Processing time (s)		Speed-up (times)	FPGA frequency (MHz)
		CPU	FPGA		
16	16 × 16	2.24	0.29	7.72	172
	32 × 32	35.91	3.47	10.35	169
	64 × 64	574.12	62.87	9.13	164
	128 × 128	9168.81	799.20	11.47	171
32	16 × 16	4.68	0.69	6.76	125
	32 × 32	71.20	5.58	12.76	118
	64 × 64	1151.89	91.43	12.60	120

Fig. 36

Table 2 FPGA resource usage versus node size

Trotter size	Node size	Resource usage				
		Logic (%)	Registers	Memory	RAM blocks (%)	DSPs (%)
16	16 × 16	127,908 (30%)	183,284	1.1 (17%)	967 (36%)	96 (6%)
	32 × 32	130,485 (31%)	183,782	1.2 (18%)	984 (36%)	96 (6%)
	64 × 64	133,584 (31%)	185,403	1.5 (22%)	1096 (40%)	96 (6%)
	128 × 128	136,621 (32%)	187,095	2.4 (36%)	1581 (58%)	96 (6%)
32	16 × 16	229,147 (54%)	264,256	2.1 (14%)	1807 (67%)	192 (13%)
	32 × 32	238,468 (56%)	266,394	2.6 (39%)	1930 (71%)	192 (13%)
	64 × 64	247,987 (58%)	268,896	3.1 (46%)	2147 (79%)	192 (13%)

Fig. 37

Table 3 Processing time versus Trotter size (500 iterations)

Node size	Trotter size	Processing time (s)		Speed-up (times)	FPGA clock frequency (MHz)
		CPU	FPGA		
32 × 32	8	18.05	2.29	7.85	250
	16	35.91	3.47	10.35	169
	32	71.20	5.58	12.76	118
64 × 64	8	285.83	33.49	8.53	253
	16	574.12	62.87	9.13	164
	32	1151.89	91.43	12.59	120

Fig. 38

下面幾張圖是我們的 result，由於沒有 source code，但是我們可以做到跟他們 comparable 的結果，也就是 optimized 後的結果。

可以看到在 size 較大的情況下，我們快了一些，將近 30 秒，那大可以歸功於我們的 frequency 較高，由於我們沒有他們的 source code，所以也可能是我們的 code 哪邊採取了較好的方法。而小 size 的情況下，我們小輸了他們的 work，可能是 implement 方式不同，因為在小 size 的情況下，coupling matrix 是可以整個放在 on-chip memory 上，而不需要從外部 memory load 近

來，所以推測他們的 coupling matrix 是放在 on-chip memory 上，因此在小 size 的 case 下比我們快了一些。

Trotter size	Node size	FPGA	MHz
16	16x16	0.418	166
	32x32	2.979	190
	64x64	45.0915	188
	128x128	767.921	176
32	16x16	0.472	145
	32x32	3.9095	151
	64x64	61.331	140

Fig. 39

Trotter size	Node size	Resource usage			
		ALUTs	FFs	RAMs	DSPs
16	16x16	11%	9%	37%	3%
	32x32	11%	9%	31%	3%
	64x64	12%	11%	40%	3%
	128x128	12%	16%	56%	3%
32	16x16	22%	17%	71%	5%
	32x32	22%	18%	71%	5%
	64x64	22%	21%	77%	5%

Fig. 40

Node size	Trotter size	FPGA	MHz
32x32	8	2.6365	214
	16	2.979	190
	32	3.9095	151
64x64	8	36.1225	234
	16	45.0915	188
	32	61.331	140
128x128	16	767.921	176

Fig. 41

下圖 Fig. 42 是說明使用了 SQA 確實可以達到 SA 達不到的 low energy。

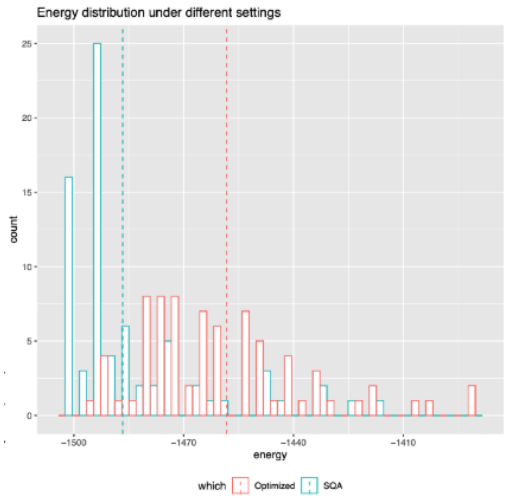
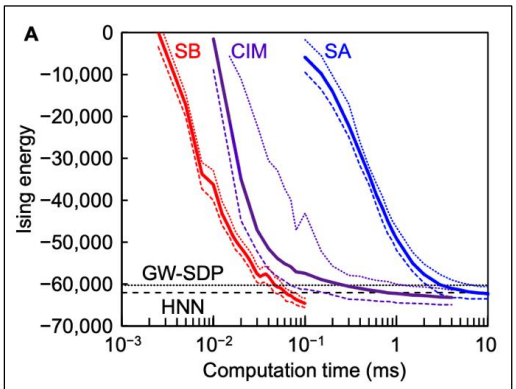


Fig. 42

再來是講到 Toshiba 在 SB 上面的 work，我們也做了相對應的實驗。

下圖是他們的 work，實驗的問題是 K2000，是一個 max-cut problem 轉化出來的 Ising problem，是一個 2000x2000 fully-connected 的 ising problem。

圖表 Fig. 43 是達到 GW-SDP 或 HNN 的 energy 所需要的時間，他們是跑在 Arria 10 gx FPGA 上面的結果，由於整個計算幾乎是在做矩陣相乘，可以在 FPGA 上做得非常快，是 SA 的 update 方法無法相比的。



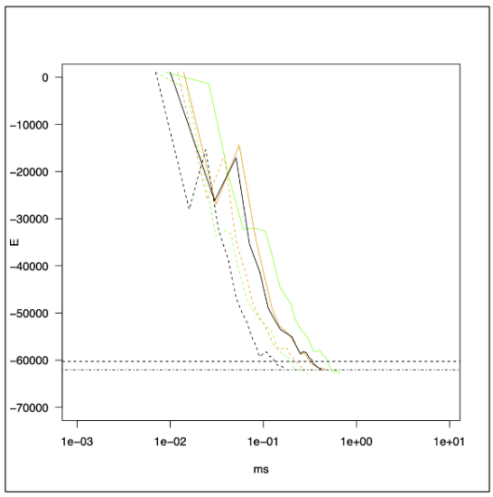
	Ave. of HNN			GW-SDP		
	Best (ms)	Ave. (ms)	Worst (ms)	Best (ms)	Ave. (ms)	Worst (ms)
SB	0.047	0.061	0.074	0.040	0.047	0.058
CIM	0.155	0.769	N/A	0.071	0.264	1.16
SA	2.64	6.80	N/A	2.10	3.20	7.15

Fuig. 43

下圖 Fig. 44 是我們相對應的 work，我們是做在 GPU 上面，可以想像不像 FPGA 整個拿來做 SB 那麼快，但是相比起來，速度應該也是不差的，雖然比他們 FPGA SB 慢一些，但是比 CIM 和 SA 還要快。

再來的好處是 GPU 應該是相對好 program 的，且 scale 起來比 FPGA 好，價格相對於 FPGA 低。

可以看到我們使用了 cublas library，這 optimized 的矩陣乘法，讓我們能在 GPU 上發揮最大效能。



	HNN			GW-SDP		
	Best	Ave	Worst	Best	Ave	Worst
Naive	0.234	0.344	0.608	0.192	0.266	0.507
Modify	0.211	0.288	0.451	0.183	0.238	0.370
cuBLAS	0.149	0.222	0.427	0.112	0.180	0.362

Fig. 44

下圖 Fig. 45 是我們更進一步增加 problem size，在我們的 GPU 上(RTX 2080 Ti 和 V100)，幾乎做一輪所有 node 的更新，所要花的時間是非常小的。單張 GPU 可以 scale 到 90112 fully-connected Ising problem 都可以解的非常快，這應該是那單張 FPGA 做不到的事情。

RTX 2080 Ti 11GB	Update all node in one step (divided by 100 steps)
512	5.62 μs
1024	5.69 μs
2048	5.73 μs
4096	5.95 μs
8192	6.25 μs
16384	6.43 μs
32768	6.35 μs

Tesla V100 32GB	Update all node in one step (divided by 100 steps)
512	6.65 μs
1024	7.05 μs
2048	7.97 μs
4096	7.13 μs
8192	8.16 μs
16384	8.69 μs
32768	8.80 μs
65536 (using MM)	12.17 μs
90112 (using MM)	12.45 μs

Fig. 45

Appendix

CIM simulation on classical computer

在 2018 年的時候，D-wave 在 arxiv 上發了一篇 CIM 的 simulation 的論文，" Emulating the coherent Ising machine with a mean-field algorithm"。以 CIM 是將 Ising problem 的能量最小化，其中在前面推演其 Hamiltonian 的過程可以發現，整個 CIM 降低能量的過程是會受到 noise 所干擾的。而 D-wave 發表的方式則是利用與 CIM 類似的 noise，進行比較平均場退火的演算法，Noisy mean-field annealing (NMFA) algorithm。Pseudo code 如下所示：

Algorithm 1 Noisy mean-field annealing. Generates a set of Ising spins $\tilde{s}_i$ given Ising problem (h, J) and parameters T, σ, and α.

for $i = 1$ to N **do**
  $s_i := 0$
end for
for $t = 1$ to t_f **do**
  for $i = 1$ to N **do**
    $\Phi_i := (h_i + \sum_j J_{ij} s_j) / \sqrt{h_i^2 + \sum_j J_{ij}^2} + \mathcal{N}(0, \sigma)$
    $\hat{s}_i := -\tanh(\Phi_i / T_t)$
  end for
  for $i = 1$ to N **do**
    $s_i := \alpha \hat{s}_i + (1 - \alpha) s_i$
  end for
end for
for $i = 1$ to N **do**
  $\tilde{s}_i := s_i / |s_i|$
end for

從演算法上講，CIM 遵循一個簡單的循環，也就是我們在前面所提到的 CIM 的一個 circulation，在每次的循環中重複測量 spin 的 state，然後跟從該測量中得出的 mean-field 做組合，去決定是不是要做 coupling 的動作。D-wave 將 CIM 性能與 NMFA 進行了比較，NMFA 是一種演算法，使用[-1，1]，而不是 CIM 中所使用的光脈衝相位。CIM 使用光脈衝注入 mean-field，而 NMFA 使用 Boltzmann factor 注入 mean-field 用以降低溫度，作法類似 annealing (相類似於 SA 之類的做法)。而在此篇論文中，D-wave 也有敘述了 NMFA 演算法中的變數對應於 CIM 中的變數以及相對應的物理現象，在這篇報告不會詳細做討論。其中 D-wave 使用跑在 GPU 上的 code 也有 release 出來。

D-wave 將 NMFA 演算法跑在 GPU 上，而 CIM 則是跑在 Stanford 的 CIM 上。並且最後得到的結果如下表格與圖示：

TABLE I. **Mean and best performance (out of 100 runs) on 2000-spin MAX-CUT instances detailed in Ref. [2].** Values given are for the maximization problem (MAX-CUT) rather than the equivalent Ising minimization problem.

Instance	$G22$ (random)	$G39$ (scale-free)	K_{2000} (fully-connected)
NTT CIM [2]	mean 13248, best 13313	mean 2328, best 2361	mean 32457, best 33191
NMFA	mean 13267, best 13325	mean 2339, best 2369	mean 32730, best 33186

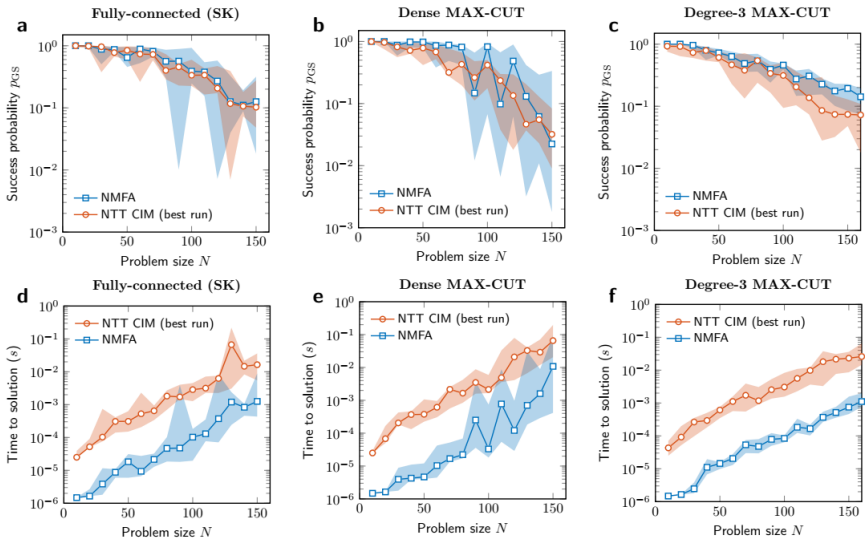


FIG. 4. **Performance of NTT CIM and NMFA on problem sets studied in Ref. [6].** Marks and shaded regions represent medians and interquartile ranges. **a–c**, success probabilities on (**a**) fully-connected Sherrington-Kirkpatrick problems with $J_{ij} \in \{-1, 1\}$, 10 instances per size; (**b**) dense MAX-CUT problems (edge probability $p = 0.5$), 10 instances per size; (**c**) degree-3 MAX-CUT problems, 20 instances per size. **d–f**, time to solution. NMFA time per sample is computed via wall-clock time across 10,000 samples; CIM time per sample is computed using 5 ms sample time for 2000 spins, divided by the number of copies of an instance that can be run in parallel.

D-wave 參考了 CIM 團隊發表的其它幾篇論文，去採用 Stanford 的 CIM 和 NTT 的 CIM 跑的三種隨機問題的結果分別是 Sherrington-Kirkpatrick（SK model）、MAX-CUT on dense graph、MX-CUT on sparse graph，並用 NMFA 也跑了相同的題目。他們使用 NTT 的 CIM 機器和 NMFA 跑在 GPU 上做比較，三種實驗的數據結果分別如上圖所示。

在圖 4a 中，進行的是 SK model 的問題，且定義 Jij = 1 的機率為 p=1/2，除此以外的 Jij=-1。圖 4b 則是跑 dense MAX-CUT 的問題，設定 Jij=1 的機率 p=1/2，剩下的 Jij=0。而圖 4c 則是 degree = 3 的 MAX-CUT，其中每個 spin 與其他三個 spin 做 coupling(所有 nonzero couplers 的 Jij = 1)。

圖 4d-4f 顯示了利用時間做為參考的實驗效能結果。

其中 D-wave 在使用 Stanford 的 CIM 跑了 10,000 次，並且取其中最佳的 1000 次結果做圖。而在 NMFA 的部分，則是 10,000 次數據全取下去做圖。

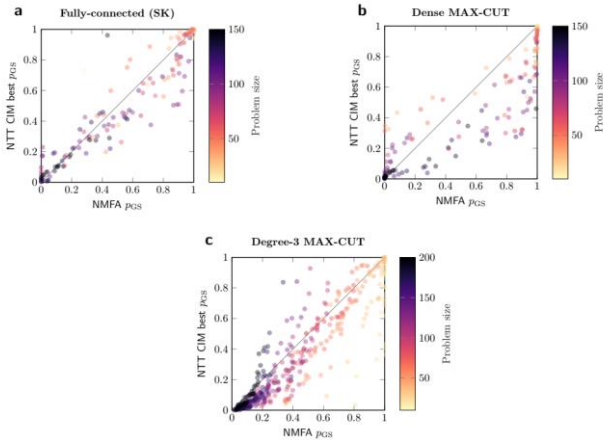


FIG. 5. **Instance-wise comparison of NTT CIM and NMFA.** We compare success probabilities for each instance studied in Fig. 4 for (**a**) SK, (**b**) dense MAX-CUT, and (**c**) degree-3 MAX-CUT problems.

在上圖中，D-wave 針對三種問題跑了 Stanford 和 NTT 的 CIM、NMFA 在 NVIDIA GeForce GTX 1080 Ti GPU 上，並計算每一次的運行時間。

跑 dense MAX-CUT 的結果是：Stanford 的 CIM((最大的 spin 數量為 N = 100)單次運行需要 1600μs。NTT 的 CIM(最大的 spin 數量為 N = 2000)的運行時間為 5000 μs，然而，因為 NTT 的 CIM 可以做平行化，因此在跑 dense MAX-CUT 的問題的時候是分成 20 個平行化，且每個 N=100，則可以將每一次的運行時間降低成 250 μs。其中，Stanford 和 NTT 的 CIM 機器的一些 overhead，像是 readout 和 postselection 都是被忽略不計的。對於在 NMFA 跑在 NVIDIA GeForce GTX 1080 Ti GPU 的結果是，每次運行時間大約是 12.3 μs。

因此最後的結果是，跑在 GPU 上的 NMFA 相對 Stanford 的 CIM 快了 130 倍。相對 NTT 的 CIM，NMFA 的速度約比其快了將近 20 倍速度。

並且在最後的圖表中可以發現，NMFA 的效能均與 CIM 相當、甚至是比不論是 NTT 還是 Stanford 的 CIM 的效能都來的更好。

在經過上面的比較後，D-wave 對 CIM 這種量子光學儀器下的評論是：儘管它確實代表了光學的一種新穎用途，但尚未提出宏觀的計算量子模型。因此，目前尚不清楚潛在的潛力相干的伊辛機具有實用性，但可作為未來的計算技術。

CIM 的實體機器與 D-wave 的實體機器的實驗比較

在 2018 年 D-wave 發表了前一個部分所提到的 CIM 的 simulation 並做出一番見解，認為 CIM 的實體機器技術還有待發展後。CIM 便於 2019 與 2020 發表了兩篇論文回應了 D-wave 對於 CIM 的評論。分別是：(2019_Science) Experimental investigation of performance differences between coherent Ising machines and a quantum annealer，與(2020_SPIE) Synchronously-pumped OPO coherent Ising machine : benchmarking and prospects。

其中，比較的對象並不是 D-wave 之前做的 NMFA 演算法，而是 D-wave 實體機器的對決。比較用的機器一樣式 Stanford 和 NTT 的 CIM 機器，與 D-wave 在 2019 最新的機器 D-wave 2000 qubits(簡稱 DW2Q)做比較。並提出 DW2Q 機器本身一個重要的問題點就是 embedding 的問題。並且針對 embedding 做了一番論述。其中包括 D-wave 的機器無法跑到更多的 qubits 的原因便是因為 D-wave 的機器在跑 fully connected 的問題的時候，會於到 embedding 的問題。因此基本上雖然有 2000 個 qubits，然而實際上只能到跑兩千開根號，大約是 64，但實際上則為 61 的 qubits。比 Stanford 的 qubits 數量 = 100，和 NTT 的 CIM 的 qubits = 2000 都來的小許多。因此在最後比較當 qubits 數目超過 61 時，D-wave 的所花的時間並不是真正使用 D-wave 機器的時間，因為他還沒辦法跑那麼多 qubits，而是根據在 N=61 以下的趨勢線去做內插所得到的結果。並且所有跑的問題皆和上面一樣。包括 SK-model、MAX-CUT on dense graph、MAX-CUT on sparse graph。

首先先看 SK-model on a fully connected graph(NP-hard)，CIM 的團隊利用 SK-model 這個問題去找到在不同的 size(也就是 N)和不同的問題下，對於 D-wave 最佳的 Jij。並在下一個實驗 MAX-XUT on dense graph 上，CIM 和 D-wave 都是使用對於 D-wave 比較有利的 Jij 的值寫入 Hamiltonian。原因為了做到讓 D-wave 的機器能夠有比較好的效能。其中 SK-model 的規定如下：

1. Couplings Jij = ±1 發生機率相同，隨機取。
2. Ground-stat computation of the model 和 "graph partitioning problem(NP-hard)"有關。
3. DW2Q 每輪會做 20 次，從 Problem size : $2 \leq N \leq 61$ 中隨取 N 的大小。
4. We consider as a performance metric the success probability P, defined as the fraction of runs on the same instance that return the ground-state energy
5. 在 $N \geq 60$ 的時候，此實驗只有 CIM 會進行，因為 problem size 太大，沒辦法 embed 到 DW2Q 上。而就像前面提到的，D-wave 的時間將會從已經有的在 N 比較小的實驗數據中做內插法得到。

6. 關於 SK-model 的 ground-state：

   a. SK ground states were found with the Spin Glass Server, which uses BiqMac, an exact branch-and-bound algorithm.

   → @N≤100 時，此演算法已被證明能找到 SK-model 的 ground state

   → @ N≥100 時，尚未被證實。然而在此實驗室假設在 N≤150N≤150 時，用此演算法求得之 SK ground states 都是正確的。

   ⟹ 原因：

   a.）他們在在 100≤N≤150100≤N≤150 重複跑了很多次這個演算法，去求其 SK ground states energy，求得數字都相同。

   b.)且除此之外，在實驗中 run SK-model 的時候，所得到的 SK ground states energy 皆比用此演算法求得得還高，故信之。

在得到在不同的 N 下對於 DW2Q 的最佳 Jij 後，CIM 的團隊便用了這些 based on 不同的 N 上的 Jij run 在 MAX-XUT on dense graph 的實驗上。其中此實驗的規定如下：

1. Dense and sparse for random unweighted graphs.
2. Expressed by Ising problem, settings: antiferromagnetic couplings Jij = +1
3. Problem sizes：
   (a) DW2QN ≤ 61，(b) CIM N ≤ 150
4. Finding ground-states ：
   @ N≤30，在個人筆電上暴力算出來的。
   @ 20≤N≤150，使用 Branch-and-bound algorithm.

並得到最後的結果如下圖：

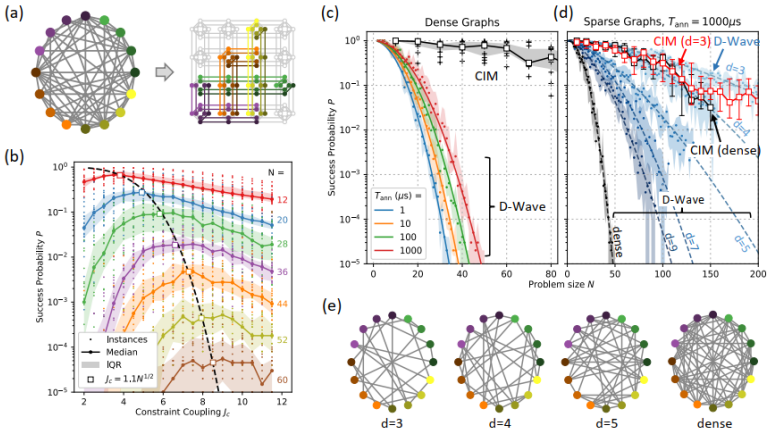


Figure 2. (a) Optimized quantum annealing of embedded SK problems on DW2Q: clique embedding to map a densely-connected problem onto a Chimera graph. (b) SK success probability as a function of problem size N and embedding constraint J_c. (c) MAX-CUT success probability for dense (edge-density-$\frac{1}{2}$) graphs. (d) Regular sparse graphs with $d = 3, 4, 5, 7, 9$ edges per vertex. (e) Representative sample of $N = 16$ sparse and dense graphs.

我們可以從上圖中發現：

1. 圖 b,D-Wave ground-state probability 在 SK model 中，調整 Jc(也就是 Hamiltonian 裡面的 Jij)做出的圖。並且將 N 和 optimal Jc 的寫成 function 供下一個實驗使用。(且固定 annealing time ，Tann=20 ms)，並求出 N 和最佳 Jij 的公式：$Jc = 1.1N^{1/2}$。
2. 圖 c，是 MAX-CUT on dense graph 的結果。其中 edge density = 1/2。且 CIM 的 data fitting curve: $P=e^{-(N/NCIM)}$，其中 NCIM 是一個上升很慢的常數。D-wave 的 data fitting curve: $P=e^{-(N/Ndw)^{-2}}$，NdwNdw 是一個上升很慢的常數，且和 T_{anne} 有關。自此結果可以看出，即使使用 optimal J_c。然而，隨著 N 的上升，D-wave 的機器的實驗 success probability 仍會以 exp 速率遞減。另外，對比上面 D-wave 的結果，還可發現 CIM 比較不會受到 N 的影響，一直到 N ≥50 後，CIM 的 success probability 才會開始急遽下降。
3. 圖 d，MAX-CUT on sparse graphs，d=3,4,5,7,9 edges/vertex。在這裡他們使用了"embedding heuristics (provided by D-wave API)"一種做 embed 的方法，他可以讓我們在 embed sparse graphs 時，只需使用較少的 physical qubits。用這個方法使 D-wave 的 constraint on connectivity 減少，進而擁有較高的 embedding overhead。從圖中可發現：
   (a) 在 d = 3(cubic), d = 4 下，DW2Q 的表現比 CIM 好。且在 cubic 的狀態下，problems size N 可調升最大至 200，embedded in the DW2Q。
   (b) 然而，在 higher density，D-wave 就會回到像在 dense graph 上一樣，success probability 高速下降。而 d = 5 時，是 CIM 效能比 DW2Q 好最顯著的 degree。
   (c) CIM 不會受到 edge density 的影響。簡而言之，就是再一次去證明，physical annealers 會受到 coupling constraints，也就是 edge density 的影響，且在 N ≥ 40 的時候，CIM 的效能會明顯優於 DW2Q。

小結論，最後比較 CIM 跟 Dw2Q 的 performance 後會得到，在 N ≥ 40 時。For MAX-CUT，在 N = 55, CIM outperforms D-Wave for factor = 10^7；在 N=100，CIM 的 outperform factor exceeds 10^{20}。簡單來說，在 N 越來越大的時後，DW2Q 的效能明顯就會比 CIM 差很多。

以上實驗的結果是，CIM 於 2020 年發表在 SPIE 上的實驗結果，而由於 CIM 在 2019 發表在 Science 上的實驗相當類似於 2020 年的實驗。只是改變 MAX-CUT 的 degree 的數量，故這裡不再花版面去放圖並闡述 2019 年的實驗結果。

並且可將所有的實驗結果整理成如下表所示：

Table 1. Time to solution T_{soln} for SK, dense MAX-CUT, and $d = 3$ MAX-CUT problems on D-Wave and NTT CIM (see section S2). The annealing time for D-Wave runs was chosen (in the range [1, 1000]μs) to optimize T_{soln} (see section S4). All CIM data are for fixed anneal times (1000 round trips). "Factor" refers to the ratio of solution times $T_{soln}^{(DW)}/T_{soln}^{(CIM)}$.

SK				MAX-CUT (dense)				MAX-CUT ($d = 3$)			
N	DW2Q	CIM	Factor	*N*	DW2Q	CIM	Factor	*N*	DW2Q	CIM	Factor
10	6.0 μs	25 μs	0.2	10	6.0 μs	25 μs	0.2	10	1.0 μs	50 μs	0.02
20	35 μs	100 μs	0.3	20	0.4 ms	100 μs	4	20	3.0 μs	100 μs	0.03
40	6.1 ms	0.4 ms	15	40	6.1 s	0.4 ms	10^4	50	12 μs	0.4 ms	0.03
60	1.4 s	0.6 ms	2000	55	10^4 s	1.2 ms	10^7	100	100 μs	3.3 ms	0.03
80*	(400 s)	1.8 ms	(10^5)	80*	(10^{11} s)	1.8 ms	(10^{13})	150	2.8 ms	22 ms	0.1
100*	(10^5 s)	3.0 ms	(10^7)	100*	(10^{19} s)	2.3 ms	(10^{21})	200	11 ms	51 ms	0.2

*D-Wave solution times extrapolated using $P = e^{(N/N_0)^2}$ fits in Figs. 2C and 3B. Note that dense problems with $N > 61$ are not embeddable in the DW2Q.

小總結

從 2003 年的 paper 開始一直看到 2020 年。從一開 CIM 團隊在做物理意義上的推導、一直到後來小機器的模擬、實體機器的出現與實體機器的使用。並在最後最了不同問題的應用。

從報告的一開始先提到了為甚麼選擇 CIM 作為此次報告的主題、再來是很簡略的先介紹甚麼是 CIM，以及 CIM 所使用的 Ising spin 是什麼。再來，對 CIM 的物理機器做詳細的介紹、接著是 CIM 的 Hamiltonian 與作用理論。便對 CIM 做了一個相對詳盡的介紹。

而在介紹完 CIM 的本體機器與作用理論後，提到 D-wave 在 2018 年使用了 NMFA 這種演算法作為 CIM simulation on classical computer，去和 CIM 的實體機器做效能的比較。會在這裡發現到，跑在傳統電腦上的 simulation，也就是 NMFA 跑在 NVIDIA GeForce GTX 1080 Ti GPU 上的效能，還是比現今實作出的量子電腦的實體機器來的好。從實驗結果可以發現，無論是從運行時間判斷其效能、還是從跑很多次所得到的結果的正確性作為運行效能的判斷基準。三種實驗，包刮 SK-model、MAX-CUT on dense graph、MAX-CUT on sparse graph 等，都是 NMFA 跑在 NVIDIA GeForce GTX 1080 Ti GPU 上的效能較佳。我們可以從這件事情上發現，量子電腦的確還有很大的空間可以討論，除了希望能夠做到降低運行時間外，另外值得討論的還有效能是否能夠提升的問題。

最後提到 CIM 的實體機器與 D-wave 的實體機器的實驗比較，分別是 CIM 的團隊於 2019、2020 年發表的兩篇 papers。從實驗結果我們不僅可以發現 CIM 相對於傳統的 annealers(像是 DW2Q)的優勢，大概分成以下兩點：

(1)首先，CIM 存在的意義則是在前面有提到的，傳統的 annealers 會因為要將 fully connected 的 Ising problem embed 過去到實體機器上，因此會浪費很多 qubit 下去做 connecting 的動作。故雖然像是 D-wave 已經可以做到有 2000 個 qubits。然而，若再遇到 degree 數量比較大的問題的時候，真正所能使用的 qubits 數量就會開始急遽下降。甚至到最後是開根號除以二的窘境。而 CIM 則是因為 OPO 產生的方式是利用 laser pump 經過 SHG 與 PPLN 去製造出我們所需要的諧振態的光波，一個 OPO 即代表一個 qubit，另外 OPO 之間並不是沒有 entanglement，詳細的解釋可以回到講解 Hamiltonian 的部分看。唯一需要擔

心的只有光無法在光纖中存在很久，因為光會消散。故，我們從 CIM 的 machine full diagram 便會發現，laser 光波會一直打回光纖中做補光的動作。當然補進去的光的 phase 和振幅肯定是有被調整過的。簡單來說，CIM 的 OPO 並不會受到 fully connected 的時後 qubits 之間需要做 entanglement 的限制。另外因為有補光的機制，因此 CIM 的 spin 數量可以一直加上去都不是問題。

(2)另一方面也證明了量子光學方法的重要性。首先 CIM 因為時做的方法不同於傳統的 annealers，因此並不會受到 coupling constraints，原因是因為 CIM 做 coupling 的方式其實只是利用光疊加的時後會改變 OPO 當下的 amplitude 與 phase 的特性，因此並不存在所謂的 coupling constraint。

總而言之，CIM 是一個不同於傳統退火機器的 Ising 機器，他是 based on 物理光學所建構出的另一種 Ising 機器。並相對一般退火機器有其自身帶來的好處。並且相待期待在未來幾年 Stanford 和 NTT 的合作，將 CIM 的機器精益求精。

Reference

1. K. Takata, Yoshihisa Yamamoto et,al. “A 16-bit Coherent Ising Machine for One-Dimensional Ring and Cubic Graph Problems.” *arXiv Quantum Physics*, 2016.

2. G. Patera, N. Treps, C. Fabre, and G. J. De Valcarcel.” Quantum theory of synchronously pumped type i optical parametric oscillators: characterization of the squeezed supermodes.” *The European Physical Journal D, 56(1):123*, 2010

3. Mater Thesis of Mar´ıa Moreno de Castro. “Control of light emission in Parametric Oscillators with Photonic Crystals” *UNIVERSITAT DE LES ILLES BALEARS*, 2009.

4. Y. Haribara, S. Utsunomiya, K-i. Kawarabayashi, and Y. Yamamoto. “A coherent Ising machine for MAX-CUT problems : Performance evaluation against semidefinite programming relaxation and simulated annealing” *arXiv Quantum Physics*, 2016.

5. Y. Haribara, Y. Yamamoto, et,al. “A coherent Ising machine with quantum measurement and feedback control.” *arXiv Quantum Physics*, 2015.

6. G. M. D’Ariano, M. G. A. Paris and M. F. Sacchi. “On the parametric approximation in quantum optics.” *ArXiv Quantum Physics*, 1999.

7. H. Takesue, T. Inagaki, K. Inaba, T. Ikuta, and T. Honjo. ” Large-scale Coherent Ising Machine” *Journal of the Physical Society of Japan 88, 061014*, 2019.

8. A. Yamamura, K. Aihara, and Y. Yamamoto. “A Quantum Model for Coherent Ising Machines: Discrete-time Measurement Feedback Formulation.” *arXiv Quantum Physics*, 2017.

9. Egor S. Tiunov, Alexander E. Ulanov, A. I. Lvovsky. “Annealing by simulating the coherent Ising machine.” *arXiv Quantum Physics*, 2019.

10. K. Takata, and Y. Yamamoto. “Data search by a coherent Ising machine based on an injection-locked laser network with gradual pumping or coupling.” *American Physical Society (APS) PHYSICAL REVIEW A covering atomic, molecular, and optical physics and quantum information*, 2014.

11. M. Yamaoka, C. Yoshimura, M. Hayashi, et,al. “A 20k-Spin Ising Chip to Solve Combinatorial Optimization Problems With CMOS Annealing” *IEEE Journal of Solid-State Circuits (Volume: 51 , Issue: 1),* 2016.

12. T. Takemoto, M. Hayashi, C. Yoshimura, M. Yamaoka. “2.6 A 2 ×30k-Spin Multichip Scalable Annealing Processor Based on a Processing-In-Memory Approach for Solving Large-Scale Combinatorial Optimization Problems”, *IEEE International Solid- State Circuits Conference - (ISSCC)*, 2019.

13. K Yang, Y.F. Chen, G. Roumpos, C. Colby, J. Anderson. “High Performance Monte Carlo Simulation of Ising Model on TPU Clusters” *arXiv Quantum Physics*, 2019.

14. M.Aramon1, G. Rosenberg, E. Valiante, T. Miyazawa, H. Tamura, H. G. Katzgraber.” Physics-Inspired Optimization for Quadratic Unconstrained Problems Using a Digital Annealer ” *Frontiers in Physisc,* 05 April 2019

15. K. Tatsumura, A. R. Dixon.,H. Goto.” FPGA-Based Simulated Bifurcation Machine” *29th International Conference on Field Programmable Logic and Applications (FPL)*, 2019.
16. F. O-Zamorano, M. A. Montemurro, S. A. Cannas, et,al. ” FPGA Hardware Acceleration of Monte Carlo Simulations for the Ising Model” *IEEE Transactions on Parallel and Distributed Systems,* 2016.
17. T. F. Rønnow, Z. Wang, J. Job, S. Boixo, S. V. Isakov, D. Wecker, J. M. Martinis, D. A. Lidar, M. Troyer. ” Defining and detecting quantum speedup”, *ArXiv Quantum Physics*, 2014.
18. Hayato Goto. “Bifurcation-based adiabatic quantum computation with a nonlinear oscillator network: Toward quantum soft computing”, *ArXiv Quantum Physics*, 2016.
19. S. Puri, C. K. Andersen, A. L. Grimsmo, A. Blais. "Quantum annealing with all-to-all connected nonlinear oscillators". *Nature Communications*, 2017.
20. H. Goto, Z. Lin, Y. Nakamura. "Boltzmann sampling from the Ising model using quantum heating of coupled nonlinear oscillators", *Scientific Reports volume 8, Article number: 7154,* 2018.
21. H. Goto, K. Tatsumura, A. R. Dixon. "Combinatorial optimization by simulating adiabatic bifurcations in nonlinear Hamiltonian systems", *Science Advances,* 19 Apr 2019.
22. J. Romero,M. Bisson, M. Fatica,M. Bernaschi. "A Performance Study of the 2D Ising Model on GPUs", *arXiv:1906.06297*, 2019.
23. J. Su, T. Tu, L. He. "A quantum annealing approach for Boolean Satisfiability problem", 5*3nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2016.
24. K. Yamamoto, W. Huang, W. Huang, S. Takamaeda, et,al., "A Time-Division Multiplexing Ising Machine on FPGAs", *HEART2017: Proceedings of the 8th International Symposium on Highly Efficient Accelerators and Reconfigurable Technologies, Article No.: 3 Pages 1–6*, 2017.