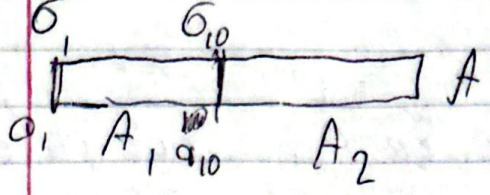


$$\sum_{a \in A_1} a = \sum_{a \in A_2} a$$

$\underline{\sigma} = \{\sigma_1, \sigma_2 \dots \sigma_n\}$

$\sigma \in \{-1, 1\}$

f_1, f_2, f_3, f_4
(1, 2, 3, 4)

We want to permute these such that

$f_{i_1} \le f_{i_2} \le f_{i_3} \le f_{i_4}$

permutation notation $\Rightarrow \pi(1,2,3,4) = i_1\ i_2\ i_3\ i_4$

Partitioning $\sum_K \sigma_K a_K = Q(\underline{\sigma}) = 0$

$\sigma_K \to +1 \Rightarrow a_K \in A_1$

$\sigma_K \to -1 \Rightarrow a_K \in A_2$

$[B, I] = sort(A)$
(B: sorted Array; I: permutation)

$B = A(I)$

$\min_{\underline{\sigma}} Q^2(\underline{\sigma}) = 0$ ← minimize energy

A's elements configured such as I.

F: $f_1, f_2, \dots f_N$

f is an array of #'s

Sorting is just finding such a permutation of indices such that the elements follow a set rule.

Sorting in some order

$S_{kj} = \begin{cases} 1 & \text{if } f_j \text{ goes to the Kth place} \\ 0 & \text{otherwise} \end{cases}$ ← defines the permutation matrix in binary form.

Sorting: f_j goes to place #m such that $f_m \le f_{m+1}$, $f_m = f_j$

S is sorting

$$\tilde{f}_k = \sum_j S_{kj} f_j$$

Such that $\tilde{f}_k \leq \tilde{f}_{k+1}$

Simply picks out one value from the array and moves it to kth place.

value of jth place goes to kth place.

If $S_{kj} = 1$ (only once for each place) then everything else is 0.

Ex:

$$\tilde{f}_{k=3} = \sum_{j=1}^{4} S_{kj} f_j$$

$f_1 \Rightarrow \tilde{f}_3$ thus $S_{kj} = 1$ if $j = 1$ else $S_{kj} = 0$

k, j, k', j' are indices

$A = 1 \Rightarrow \Sigma \Rightarrow N^2$ (some #)

$$\sum_{k,j} \sum_{k',j'} A_{k,j,k',j'} S_{kj} S_{k'j'}$$

Ex:

$$S_{k,j} S_{k'j'} = S_{1,2} S_{2,4} A_{1,2;2,4} + S_{1,2} S_{1,4} A_{1,2;1,4}$$

(not 0; = 0)

this term will be 0 since $j=2$ and $j=4$ both cannot go to $k=1$ (only one or the other, not both)

$A_{k,j,k',j'}$ = Characterizes how the penalty be applied

Ex

$$A_{1,2;2,4} = \begin{cases} 1, & \text{if } f_4 < f_2 \\ 0 & \text{else} \end{cases}$$

But A can be any #, this is just an example

What is written here is correct, but over simplified. In order to apply this extra things are needed

$A_{k,j;k',j'}$ is the out of order penalty

$$A_{k,j;k+1,j'} = \begin{cases} 1 & \text{if } f'_j < f_j \\ 0 & \text{otherwise.} \end{cases}$$

Sorting:

$$\min_{\underline{S}} \left(\sum_{k,j} \sum_{k',j'} A_{k,j;k',j'} \, S_{k,j} \, S_{k',j'} \right)$$

This works in binary $S_{k,j} \in \underline{S}$

so

$S \in \{0,1\} \Rightarrow \sigma \in \{-1,1\}$ $\underline{S}$ is permutation matrix and applying it to original list will sort it.

Binary spin

$\sigma = 2S - 1 \iff S = \frac{1}{2}(\sigma + 1)$

Sorting (spin ver.) just substitution:

$$\min_{\underline{\sigma}} \sum \ldots A_{k,j;k',j'} \, \frac{1}{2}(\sigma_{k,j}+1) \cdot \frac{1}{2}(\sigma_{k',j'}+1) \cdot$$

same as above

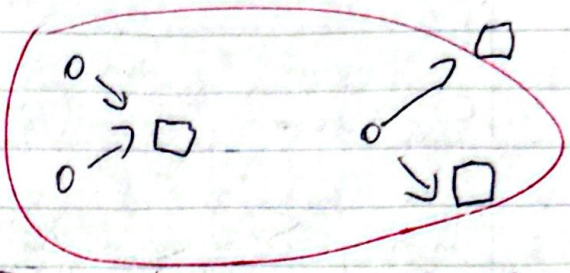
$H(\underline{\sigma})$

$\min_{\sigma} H(\underline{\sigma}) = 0$

$\arg\min_{\underline{\sigma}} H(\underline{\sigma}) = \underline{\sigma}^{(0)}$

↑ means which arg ($\underline{\sigma}$) nets the lowest $H(\underline{\sigma})$ ($H(\underline{\sigma}^0) = 0$)

Defined as so



Constraints:

Two values cannot take the same spot

One value cannot take 2 spots

$$\sum_j S_{k,j} = 1 \qquad \sum_k S_{k,j} = 1$$

↑ j only goes to 1 k

only 1 value can take up K

Formula's are states

Keyword:

Quadratic Assignment Problem (QAP)

Pseudo Boolean Optimization

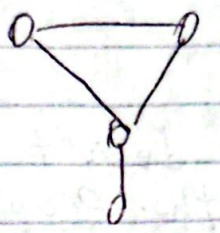
↳ Books (Hefty books) by 1965.

"Ising Machine" finds the arguments or $\underline{\sigma}$

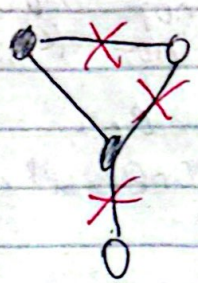
These whole idea is known as QUBO (in binary) and Ising form in Spin form.

Maximum Cut

Given a graph



and partitioned as so (dark is S_1, light is S_2)



then cut edges between nodes in different nodes. like so.

Determine the best configuration such that the maximum # of edges cut.

for $O(2^n)$ for each element added time doubles.

NP-hard is like this, scale <u>really</u> fast.

NP-complete (these are all are)

If a solution exists for NP-complete problem, then one can apply this to any NP-complete problems

NP → "all (practical) problems"

NP-hard → No efficient algorithm exists.

The time is Polynomial time.

Non-efficient → exponential w/ N 10N time.

2^N 2^{10N} $N=1000$

time increases by 2^{9000}

$2^{9000} = (10^3)^{1000} = 10^{3000}$ times more time

$T = (N=1000) = 10^{-12}$ second

For 10 N we get 10^{2988} s

NP-complete "=" NP-hard

Ising Machines ⇒ "Solve" max-cut problems

Keywords:

Heuristics – Algorithms which we do not know if they solve the problem all the time. We cannot prove that they can, and ppl do not know which ones they can.

V_2-model ← we know which problems it can solve precisely.

The ising machine can solve the partitioning problem. But no proof is shown.

Take a look at generating function

- explains the magnetism partial derivative for Z,

Domain & Domain wall for magnetism.

keywords:

FPGA

Plan:

Visually appealing version of partitioning problem/function