A Work-Time Optimal Parallel Exhaustive Search Algorithm for the QUBO and the Ising model, with GPU implementation

Masaki Tao, Koji Nakano, Yasuaki Ito, Ryota Yasudo

Hiroshima University

Masaru Tatekawa, Ryota Katsuki, Takashi Yazane, Yoko Inaba

NTT DATA Corporation

Main Contribution

- A simple exhaustive search algorithm for the quadratic unconstraint binary optimization (QUBO) problem
- It evaluates $E(X)$ for all possible 2^n n-bit input-vectors X in $O(2^n)$ time.
- We also present a work-time optimal parallel exhaustive search algorithm for the QUBO.
- It runs $O(\log n)$ time using $\frac{2^n}{\log n}$ time on the CREW-PRAM.
- Our work-time optimal parallel algorithm has been implemented on GeForce RTX 2080Ti GPU and evaluate the performance.
- We compare it with several non-exhaustive search approaches including D-Wave 2000Q quantum annealer, simulated annealing algorithm, Gurobi optimizer.

The quadratic unconstraint binary optimization (QUBO) problem

Input：symmetric matrix $W = (W_{ij})$ of size $n \times n$

Output：n-bit vector $X = x_0 x_1 \cdots x_{n-1}$

Subject to: $E(X) = \sum W_{ij} x_i x_j$ is minimized

$E(X)$ can be computed in $O(n^2)$ time in an obvious way.

→ The exhaustive search for all 2^n n-bit input-vectors X can be done in $O(2^n n^2)$ time.

→ Our sequential algorithm runs $O(2^n)$ time.

→ Since $\Omega(2^n)$ time is necessary to output all 2^n values of $E(X)$, this sequential algorithm is optimal.

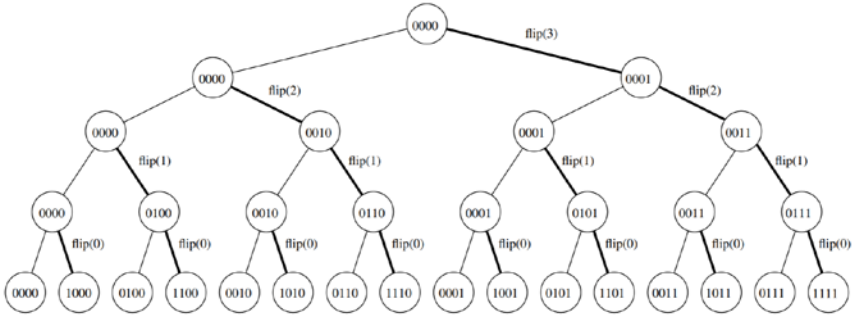
Our sequential algorithm QUBO-ES for the exhaustive search

Traverses a full binary tree with 2^n leaves

Each leaf corresponds to a value of n-bit vector X

QUBO Exhaustive Search Algorithm (QUBO-ES)

```
flip(i){
  e ← e + d_i;
  for j ← 0 to i − 1 do
    d_j ← d_j + 2W_{i,j} σ(x_i) σ(x_j);
    d_i ← −d_i ;
    x_i ← x̄_i x_i;
}
search(i){
  if(i = 0) E[f_i(X)] ← e;
  else {
    search(i − 1);  // left child
    flip(i − 1);
    search(i − 1);     // right child
    flip(i − 1);
  }
}
main(){
  for k ← 0 to n − 1
    d_k ← W_{k,k};
    X ← 00 ⋯ 0;
    e ← 0;
  search(n);
}
```



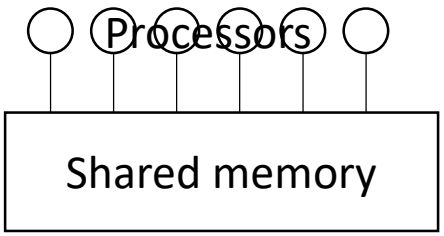
flip(i) takes $O(i)$ time

Total running time is:

$$\sum_{i=0}^{n-1} O(i) \cdot 2^{n-1-i} = O(2^n)$$

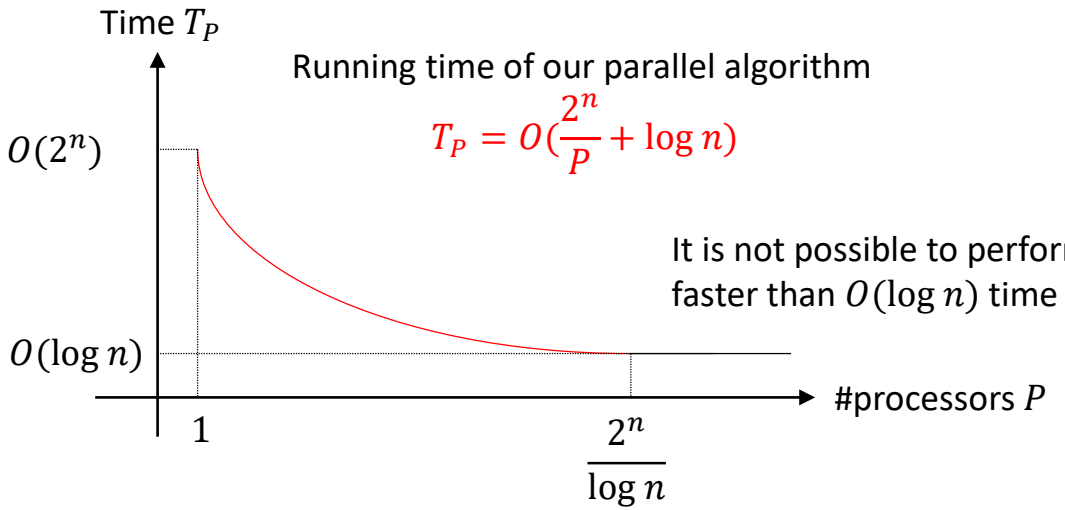
Work-time optimal parallel algorithm:

- We assume PRAM (Parallel Random Access Machine) model



PRAM

Running time of optimal sequential algorithm: $t = O(2^n)$
Running time of parallel algorithm using P processors : T_P
→ if $t = O(PT_P)$ then Work Optimal



It is not possible to perform the exhaustive search faster than $O(\log n)$ time → Time Optimal

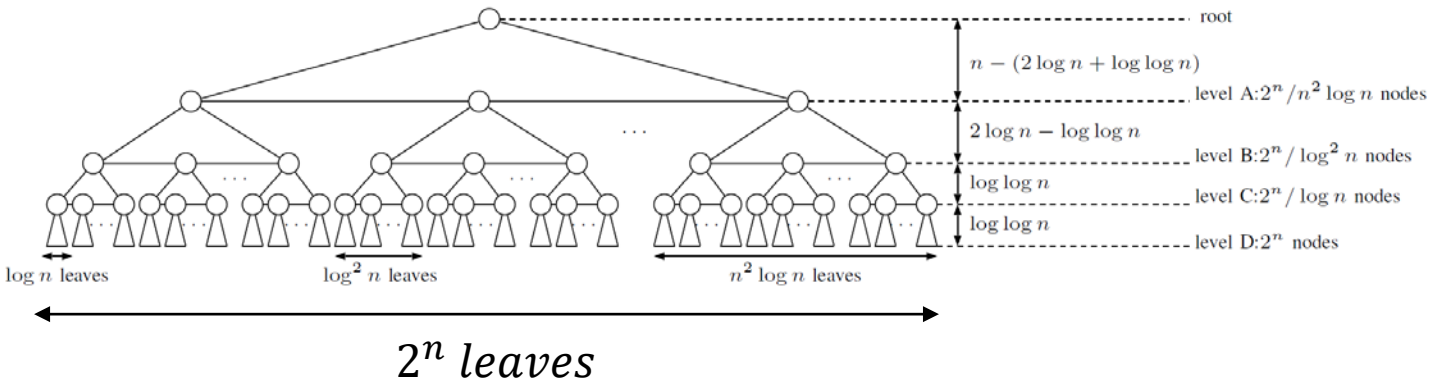
$\Omega(\log n)$ time is necessary to compute the parity of n numbers.

Work optimal and Time optimal
→ Work-Time Optimal
Theoretically, no better parallel algorithm exists

Outline of our parallel algorithm

Processors $P = \frac{2^n}{\log n}$, Time $T_p = O(\log n)$

探索木：ノードが個々の解Xに対応．各ノードの$E(X)$を全て求める．



Step 1: Search for Level A. Assign n^2 processors to each of $\frac{2^n}{n^2 \log n}$ nodes in level A

Step 2 Search for Level B. Assign $\log n$ processors to each of $\frac{2^n}{\log^2 n}$ nodes in level B

Step 3: Search for Level C. Assign 1 processor to each of $\frac{2^n}{\log n}$ nodes in level C

Step 4: Search for Level D . Assign 1 processor to a group of $\log n$ nodes in level D

Every step can be done in $O(\log n)$ time $\frac{2^n}{\log n}$ processors

Result

CPU:Intel® Core™ i7-4790
GPU:GeForce RTX 2080Ti

- RANDOM QUBO Instance

	exhaustive search			non-exhaustive search				
	QUBO-ES			D-Wave 2000Q (1000 trials)		Simulated Annealing(1000 trials)		Gurobi
n	CPU(ms)	GPU(ms)	speed-up	time(ms)	optimal count	time(ms)	optimal count	time(ms)
25	624	16	39.0	404	86	41	602	17
30	20200	38	531.6	404	81	49	1000	34
35	641894	518	1239.2	404	179	95	1000	259
40	20628753	16590	1243.4	404	519	99	747	399
45	–	525412	–	404	184	146	1000	35
50	–	17037920	–	404	43	176	1000	136

- TSP Instance

cities	n								
6	25	624	15	41.6	404	0	39	400	37
7	36	1288300	1000	1288.3	404	0	88	490	81
8	49	–	8353583	–	404	0	173	308	413

Conclusion

- We have presented GPU implementation for exhaustive search for the QUBO.
- Our GPU implementation of QUBO-ES is more than 1000 times faster than the CPU implementation whenever $\mathrm{n} \geq 33$.
- We have also shown experimental result of our QUBO-ES and non-exhaustive search approaches.