(Classical) Time-Memory Tradeoffs for Subset Sum and Decoding

Master Thesis / Quantum Information Colloquium

Alexander Kulpe

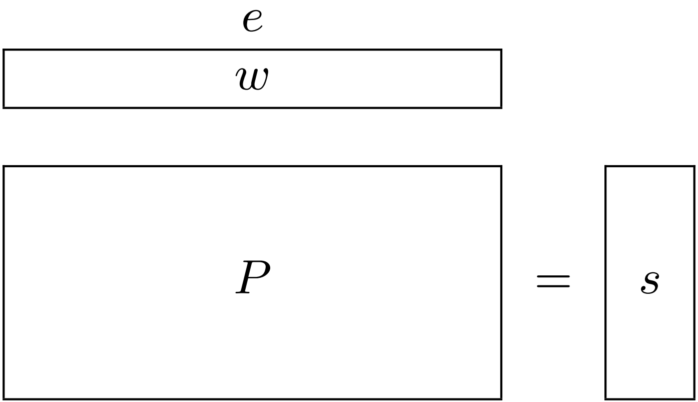
Ruhr-University Bochum
Technology Innovation Institute Abu Dhabi

2024-04-23

Table of Contents

Table of Contents

Motivation: Codebased Cryptography



- can be thought of as a vectorial subset sum variant

$\Rightarrow$ Improvements for Subset Sum might help with Decoding

Table of Contents

Subset Sum

Problem: Random Subset Sum

- **Given:** $((a_1, \ldots, a_n), t) \in (\mathbb{Z}_{2^n})^n \times (\mathbb{Z}_{2^n})^n$ with $t = \sum_{i=1}^n \varepsilon_i a_i \bmod 2^n$, $\varepsilon \in \{0,1\}^n \left(\frac{n}{2}\right)$
- **Task:** Find valid ε

- Application in Cryptanalysis / ISD algorithms
- Best algorithms are very memory-intensive

$\Rightarrow$ Time-Memory Tradeoffs

First Algorithms

Brute-Force

- **Time:** $\tilde{\mathcal{O}}(2^n)$
- **Memory:** $\tilde{\mathcal{O}}(1)$

Meet-in-the-Middle

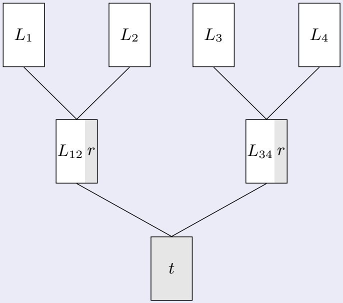
$$\sum_{i=1}^{n} \varepsilon_i a_i = t \bmod 2^n$$

$$\Leftrightarrow \sum_{i=1}^{\frac{n}{2}} \varepsilon_i a_i = t - \sum_{i=\frac{n}{2}+1}^{n} \varepsilon_i a_i \bmod 2^n$$

- **Time:** $\tilde{\mathcal{O}}\left(2^{\frac{n}{2}}\right)$
- **Memory:** $\tilde{\mathcal{O}}\left(2^{\frac{n}{2}}\right)$

Schroeppel-Shamir

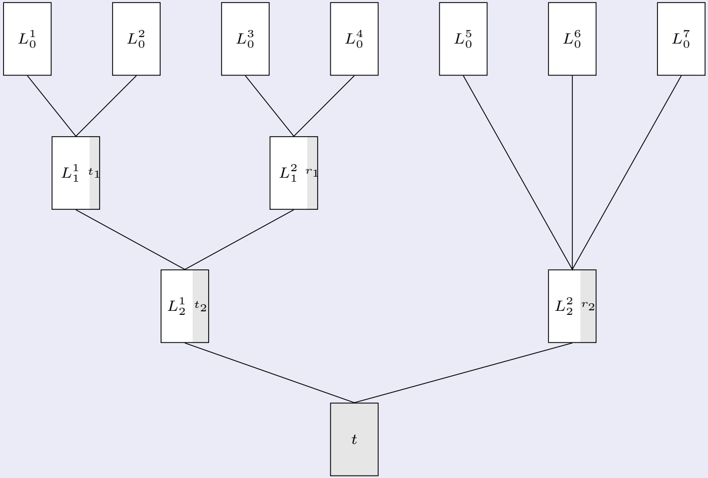
$$\underbrace{\sum_{i=1}^{\frac{n}{4}} \varepsilon_i a_i}_{L_1} + \underbrace{\sum_{i=\frac{n}{4}+1}^{\frac{n}{2}} \varepsilon_i a_i}_{L_2} = t - \underbrace{\sum_{i=\frac{n}{2}+1}^{\frac{3}{4}n} \varepsilon_i a_i}_{L_3} - \underbrace{\sum_{i=\frac{3}{4}n+1}^{n} \varepsilon_i a_i}_{L_4} \bmod 2^n$$



Time: $\tilde{\mathcal{O}}\left(2^{\frac{n}{2}}\right)$, **Memory:** $\tilde{\mathcal{O}}\left(2^{\frac{n}{4}}\right)$

First Algorithms II

7-Dissection



Time: $\tilde{\mathcal{O}}\left(2^{\frac{4}{7}n}\right)$, **Memory:** $\tilde{\mathcal{O}}\left(2^{\frac{1}{7}n}\right)$

Lemma (7-Dissection-Tradeoff)

$\frac{1}{7} \leq \lambda \leq \frac{1}{4}$. Random Subset Sum *can be solved in expected Memory* $M = \tilde{\mathcal{O}}\left(2^{\lambda n}\right)$ *and expected Time* $T = \tilde{\mathcal{O}}\left(2^{\frac{2}{3}(1-\lambda)n}\right)$.

Representation Trick

- **Idea:** Consider a larger search space with even more solutions
- Search Space MITM: $\mathcal{S} = \{0,1\}^{\frac{n}{2}} \times \{0\}^{\frac{n}{2}}$
- Search Space Representations: $\mathcal{S} = \{0,1\}^n \binom{n}{\frac{n}{4}}$
- Instead of one solution $\varepsilon \in \{0,1\}^n \binom{n}{\frac{n}{2}}$ now $\binom{n/2}{n/4}$-many representations $(\varepsilon_1, \varepsilon_2) \in \mathcal{S}^2$ with $\varepsilon = \varepsilon_1 + \varepsilon_2$

Example ($n = 8$)

- MITM: $\varepsilon = 10100110$
- Representation:

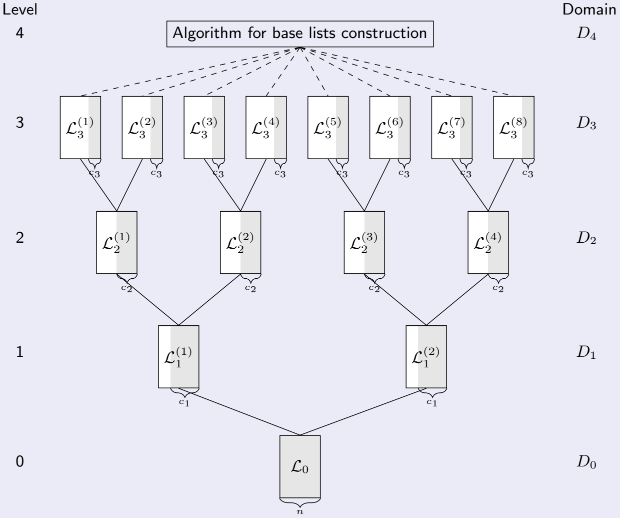
$$(10100000, 00000110) \quad (10000100, 00100010) \quad (10000010, 00100100)$$
$$(00100100, 10000010) \quad (00100010, 10000100) \quad (00000110, 10100000)$$

	MITM	Representations
$\lvert\mathcal{S}\rvert$	$2^{\frac{n}{2}}$	$\binom{n}{n/4} = 2^{0.8113n}$
$\mathbb{E}$ # Solutions	1	$\binom{n/2}{n/4} = 2^{n/2}$

$\Rightarrow$ Consider only $2^{-n/2}$-fraction of search space for a solution

HGJ

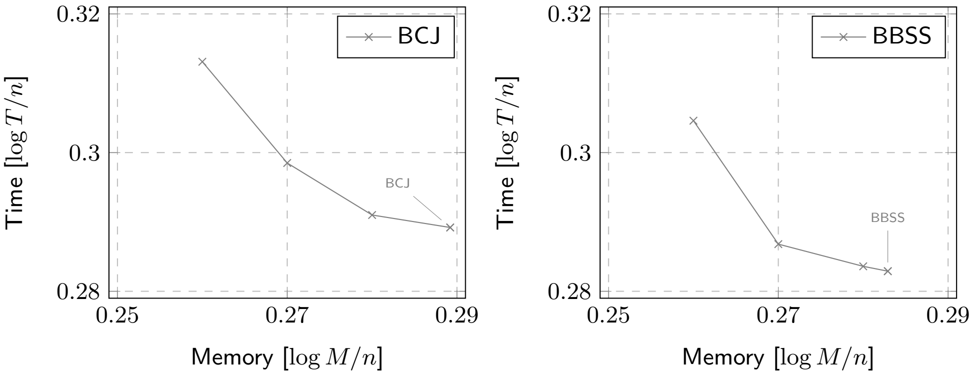
Howgrave-Graham-Joux



Rep.	T	M
$\{0,1\}$	$\tilde{\mathcal{O}}\left(2^{0.3373n}\right)$	$\tilde{\mathcal{O}}\left(2^{0.3113n}\right)$
$\{0,1,-1\}$	$\tilde{\mathcal{O}}\left(2^{0.2892n}\right)$	$\tilde{\mathcal{O}}\left(2^{0.2892n}\right)$
$\{0,1,-1,2\}$	$\tilde{\mathcal{O}}\left(2^{0.2829n}\right)$	$\tilde{\mathcal{O}}\left(2^{0.2829n}\right)$

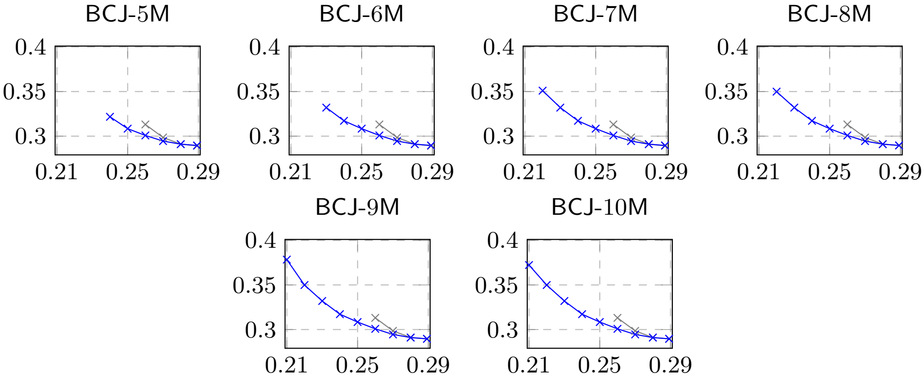
Table of Contents

Subset Sum Tradeoff: Implicit Tradeoff



- **Observation:** Higher Levels dominate Memory and Time complexity
- **Solution approaches:**
  - Increase depth of search tree
  - Swap the algorithm for base lists construction

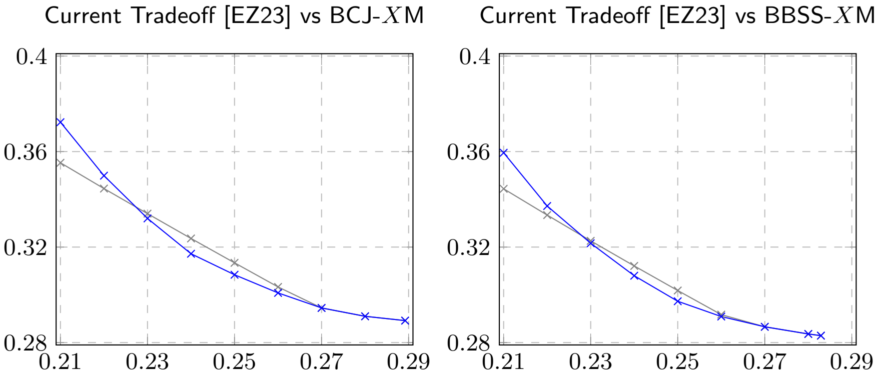
Subset Sum Tradeoff: Higher Depth I



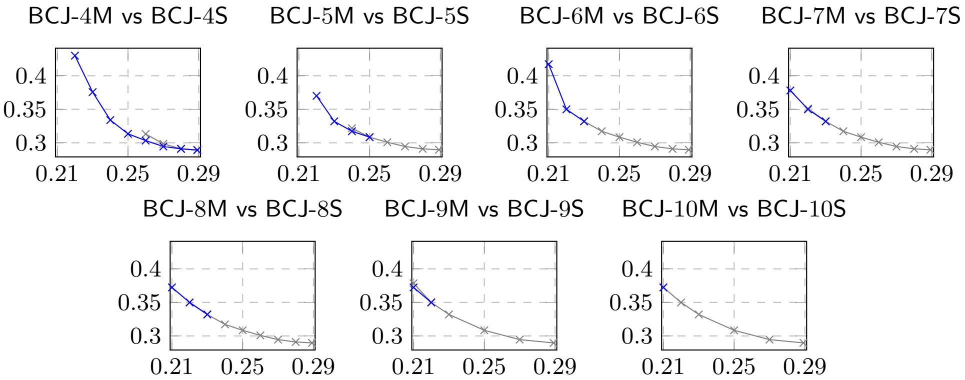
- monotonically decreasing* and convergent

*Conditions apply

Subset Sum Tradeoff: Higher Depth II



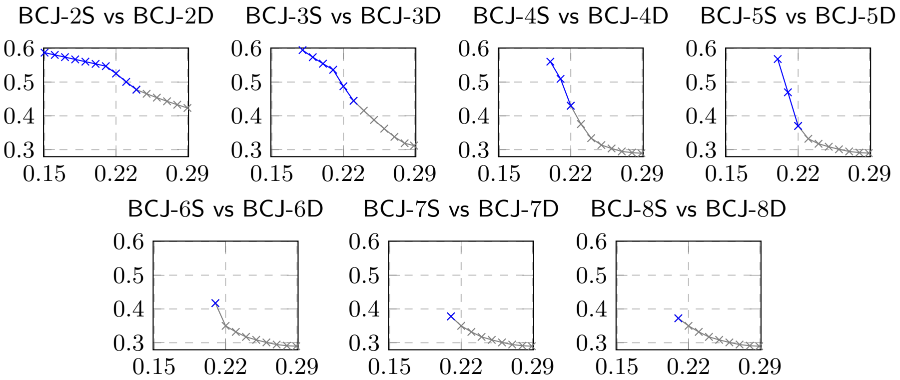
Subset Sum Tradeoff: Schroeppel-Shamir



- monotonically decreasing and convergent
- Schroeppel-Shamir for fixed depth $X < 10$ better than MITM
- BCJ-XM = BCJ-XS

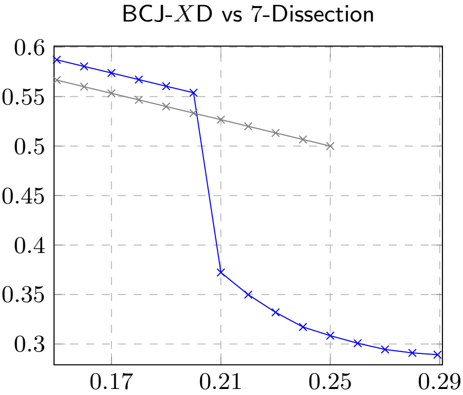
$\Rightarrow$ Depth more important than algorithm for base lists construction

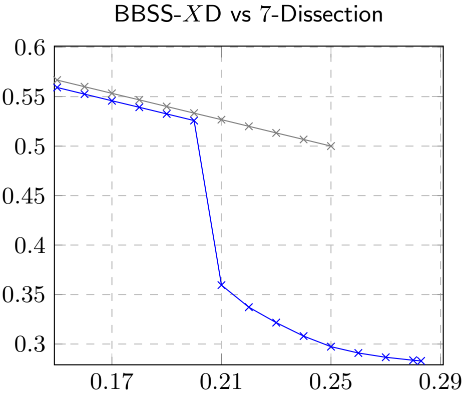
Subset Sum Tradeoff: 7-Dissection



- $\log M \geq 0.21n$: monotonically decreasing and convergent
- $\log M \leq 0.20n$: base lists construction dominates time complexity
  - $\Rightarrow$ Smaller depth better (?)

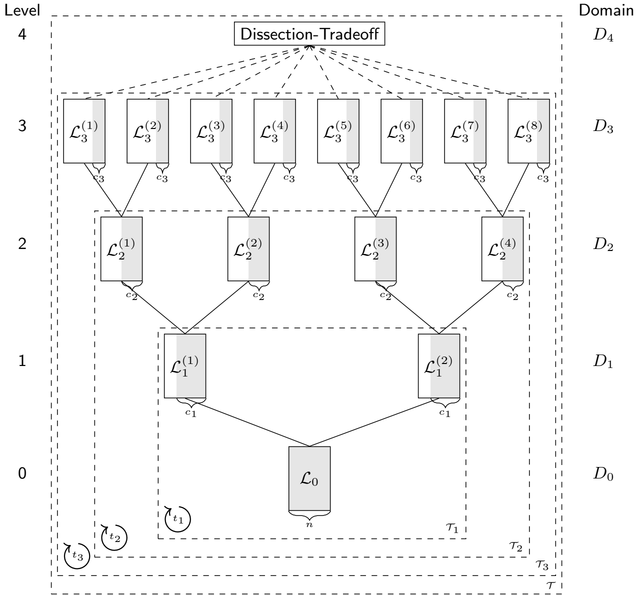
Subset Sum Tradeoff: 7-Dissection II ($\log M \leq 0.20n$)



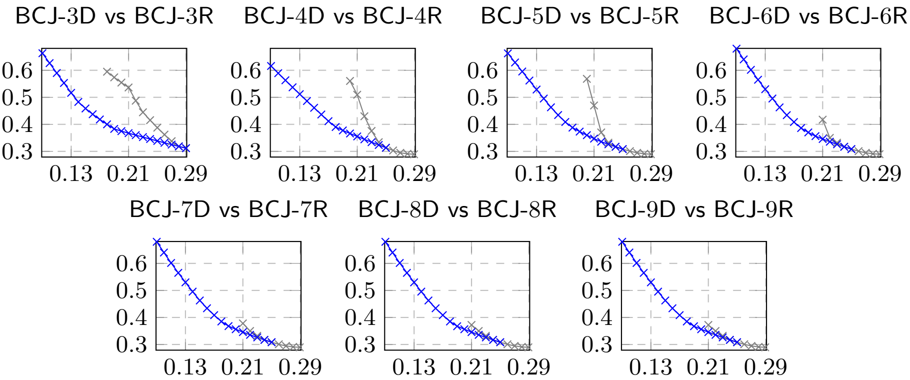


- BCJ: BCJ-XD worse than plain 7-Dissection
- BBSS: BBSS-XD better than plain 7-Dissection with optimal depth 3

Subset Sum Tradeoffs: Currently best Tradeoff / Reuse of already calculated subtrees [EZ23]

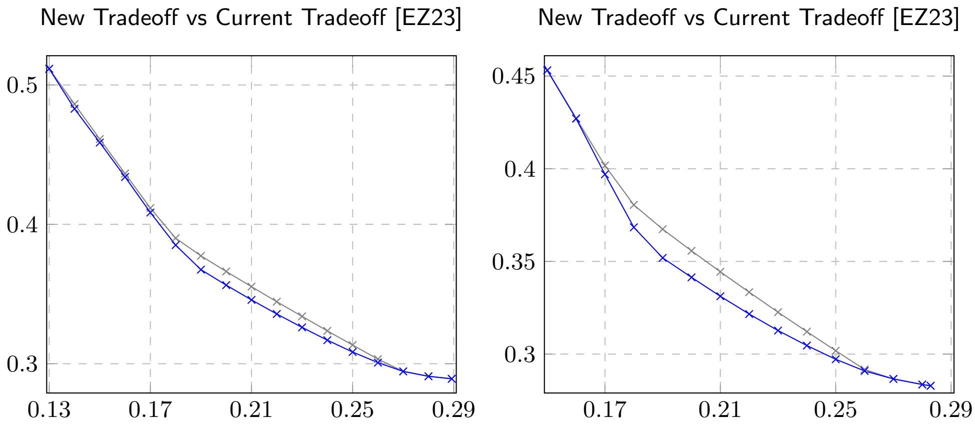


Subset Sum Tradeoffs: Reuse of already calculated subtrees



- $\log M \geq 0.19n$: monotonically decreasing and convergent
- $0.16n \leq \log M \leq 0.18n$: monotonically decreasing and convergent, lower optimal depth
- $\log M \leq 0.15n$: Base lists construction and lower lists in lower depth better balanced (BCJ: depth $3, 4$, BBSS: depth 4)

Subset Sum Tradeoff: Contribution

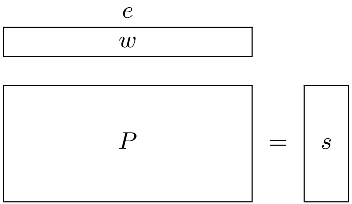


- BCJ: Improvement of up to $\tilde{\mathcal{O}}\left(2^{0.0099n}\right)$ / $2.68\ \%$
- BBSS: Improvement of up to $\tilde{\mathcal{O}}\left(2^{0.0155n}\right)$ / $4.22\ \%$

Table of Contents

Syndrome Decoding

- linear $[n, k, d]$-Code C: C is subspace of $\mathbb{F}_2^n$ with length n, dimension k and distance d
- Parity-Check-Matrix P: $C = \{c \mid c \in \mathbb{F}_2^n, Pc^t = 0\}$
- c code word, $x = c + e$ faulty codeword with error vector e
- Syndrome s: $s = Px^t = P(c^t + e^t) = Pe^t$

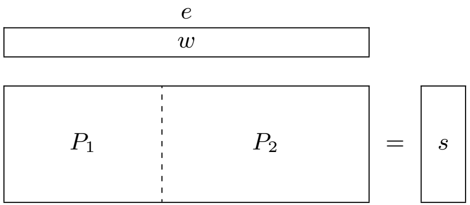


Syndrome Decoding Problem

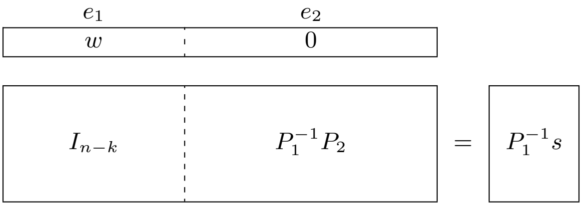
- **Given:** Parity-Check-Matrix $P \in \mathbb{F}_2^{(n-k)\times n}$, Syndrom $s \in \mathbb{F}_2^{n-k}$, Weight w
- **Task:** Find error vector $e \in \mathbb{F}_2^n(w)$ s.t. $Pe^t = s$

- half distance: $w = \lfloor \frac{d-1}{2} \rfloor$
- full distance: $w = d - 1$

Prange



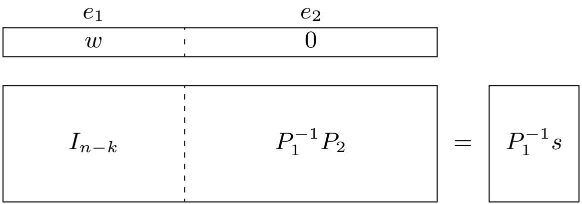
Prange



- $e_1 + P_1^{-1} P_2 e_2 = P_1^{-1} s$
- If $e_2 = 0^k$ then $e_1 = P_1^{-1} s$

$\Rightarrow$ Permute P, s.t. $\mathrm{wt}(e_1) = w$

Prange

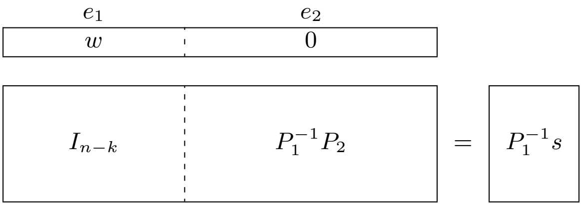


- $e_1 + P_1^{-1}P_2 e_2 = P_1^{-1}s$
- If $e_2 = 0^k$ then $e_1 = P_1^{-1}s$

$\Rightarrow$ Permute P, s.t. $\mathrm{wt}(e_1) = w$

- Time: $T = \Pr[\text{good permutation}]^{-1}$

Prange

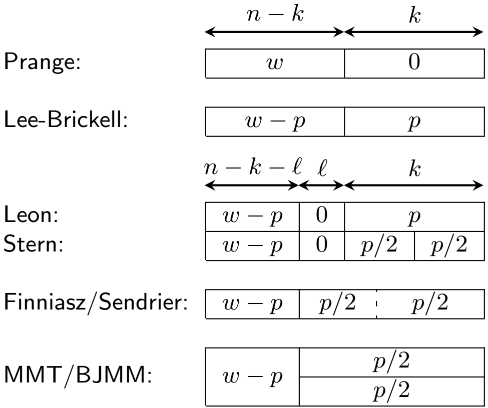


- $e_1 + P_1^{-1}P_2e_2 = P_1^{-1}s$
- If $e_2 = 0^k$ then $e_1 = P_1^{-1}s$

$\Rightarrow$ Permute P, s.t. $\mathrm{wt}(e_1) = w$

- Time: $T = \Pr[\text{good permutation}]^{-1}$
- Can we increase the probability of finding a good permutation

ISD



MMT

- Representations:

$$1 = 0 + 1$$
$$1 = 1 + 0$$
$$0 = 0 + 0$$

- optimal depth: 2

BJMM

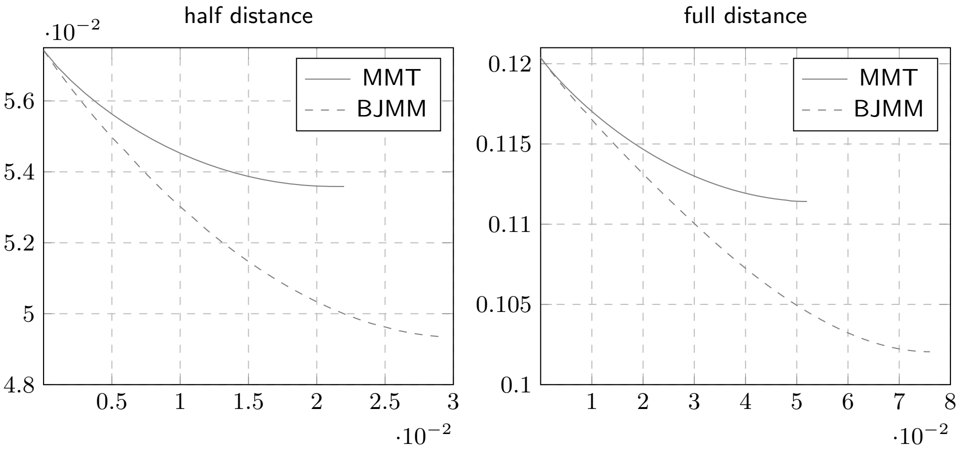
- Representations:

$$1 = 0 + 1$$
$$1 = 1 + 0$$
$$0 = 0 + 0$$
$$0 = 1 + 1$$

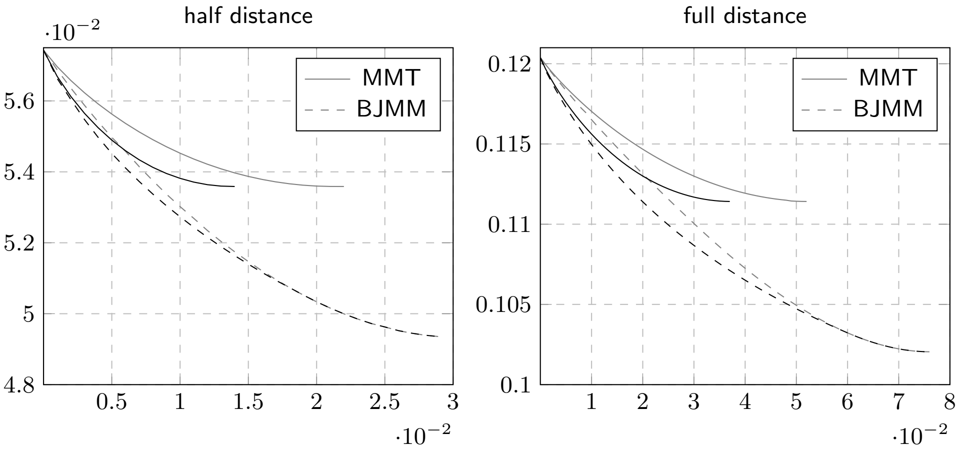
- optimal depth: 3

Table of Contents

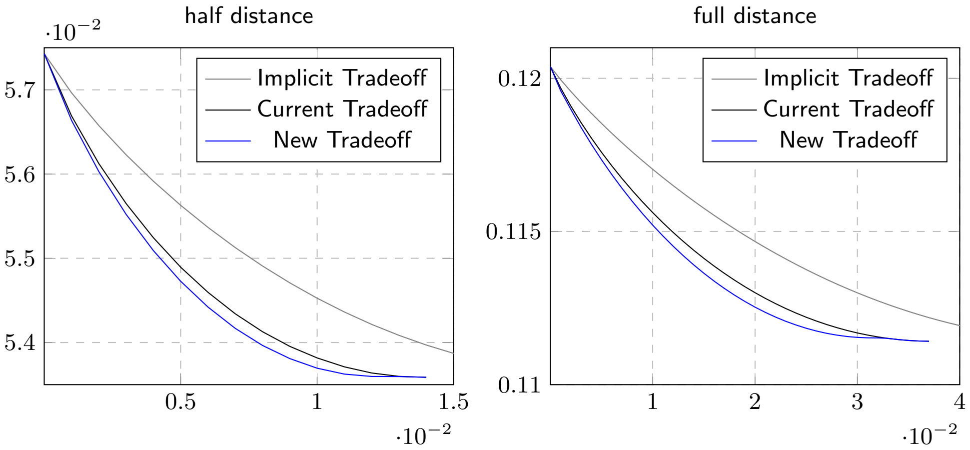
Decoding Tradeoff: Implicit Tradeoff



Decoding Tradeoff: Reuse of already calculated subtrees [EZ23]



MMT Tradeoff: Contribution



- half distance: Improvement of up to $\tilde{\mathcal{O}}\left(2^{0.000175n}\right)$ / 0.32 %
- full distance: Improvement of up to $\tilde{\mathcal{O}}\left(2^{0.000492n}\right)$ / 0.43 %
- BJMM: No Improvement

Summary / Outlook

Subset Sum

- Increase depth
- Swap algorithm for base lists construction
- Reuse already calculated subtrees
- BCJ: Improvement of up to $\tilde{\mathcal{O}}\left(2^{0.0099n}\right)$ / $2.68\ \%$
- BBSS: Improvement of up to $\tilde{\mathcal{O}}\left(2^{0.0155n}\right)$ / $4.22\ \%$

Decoding

- MMT: Improvement of up to $\tilde{\mathcal{O}}\left(2^{0.000492n}\right)$ / $0.43\ \%$
- BJMM: No Improvement

$\Rightarrow$ BJMM asymptotically better, MMT used in practice

Open Questions

- Further Applications for new Subset Sum Tradeoff
- Implementation of new MMT variant and analysis

Beware!

Optimal algorithm parameters are in general not optimal for tradeoffs!

Questions?

Optimization Scripts can be found under:
`https://github.com/alexkulpe/time-memory-tradeoffs-for-subset-sum-and-decoding`

Table of Contents

Quantum Potential

Permutations

⇒ Grover

Search for matching vectors

- can be generalized to k-list matching problem

⇒ Quantum Walks

Definition (k-list matching problem)

- **Given:** k equal sized lists $L_1, \ldots, L_k$ of binary vectors, function f that decides whether $(v_1, \ldots, v_k) \in L_1 \times \cdots \times L_k$ "match" or not (output 1 if match, 0 otherwise)
- **Find:** all k-tuples $(v_1, \ldots, v_k) \in L_1 \times \cdots \times L_k$ s.t. $f(v_1, \ldots, v_k) = 1$

Examples:

BJLM13 Combine HGJ with new data structure for quantum walks on Johnson graphs

- Reduce vertex size to get tradeoffs

BBSS20 Quantum Asymmetric HGJ: "nested" quantum search + "quantum filtering"

- Increase Asymmetry to get tradeoffs
- merging with different distributions is more difficult