Optimisation While Streaming

Amit Chakrabarti

Dartmouth College

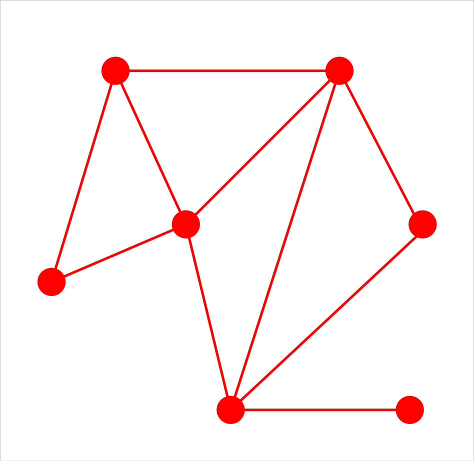
Joint work with S. Kale, A. Wirth

DIMACS Workshop on Big Data
Through the Lens of Sublinear Algorithms, Aug 2015

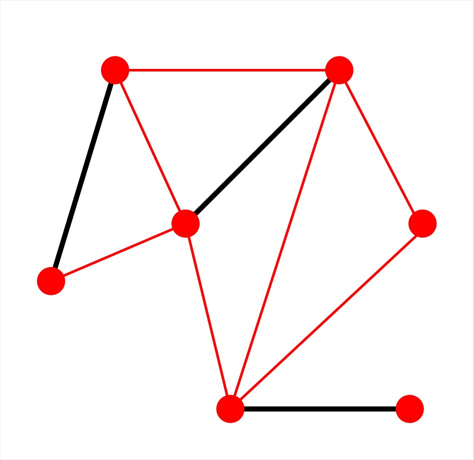
Combinatorial Optimisation Problems

- 1950s, 60s: Operations research
- 1970s, 80s: NP-hardness
- 1990s, 2000s: Approximation algorithms, hardness of approximation
- 2010s: Space-constrained settings, e.g., streaming

Maximum Matching

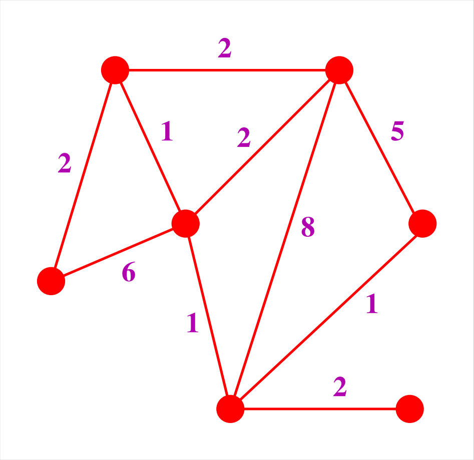


Maximum Matching

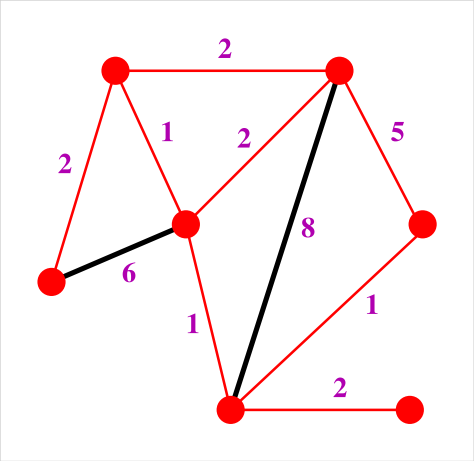


The cardinality version

Maximum Matching



Maximum Matching



The weighted version

Graph Streams: Maximum Matching, Generalisations

Maximum cardinality matching (MCM)

- Input: stream of edges $(u, v) \in [n] \times [n]$
- Describes graph $G = (V, E)$: n vertices, m edges, undirected, simple
- Each edge appears exactly once in stream
- Goal
  - Output a matching $M \subseteq E$, with $|M|$ maximal

Graph Streams: Maximum Matching, Generalisations

Maximum cardinality matching (MCM)

- Input: stream of edges $(u, v) \in [n] \times [n]$
- Describes graph $G = (V, E)$: n vertices, m edges, undirected, simple
- Each edge appears exactly once in stream
- Goal
  - Output a matching $M \subseteq E$, with $|M|$ maximal
  - Use *sublinear* (in m) working memory
  - Ideally $O(n \operatorname{polylog} n)$ ... "semi-streaming"
  - Need $\Omega(n \log n)$ to store M

Graph Streams: Maximum Matching, Generalisations

Maximum cardinality matching (MCM)

- Input: stream of edges $(u, v) \in [n] \times [n]$
- Describes graph $G = (V, E)$: n vertices, m edges, undirected, simple
- Goal: output a matching $M \subseteq E$, with $|M|$ maximal

Maximum weight matching (MWM)

- Input: stream of weighted edges $(u, v, w_{uv}) \in [n] \times [n] \times \mathbb{R}^+$
- Goal: output matching $M \subseteq E$, with $w(M) = \sum_{e \in M} w(e)$ maximal

Graph Streams: Maximum Matching, Generalisations

Maximum cardinality matching (MCM)

- Input: stream of edges $(u, v) \in [n] \times [n]$
- Describes graph $G = (V, E)$: n vertices, m edges, undirected, simple
- Goal: output a matching $M \subseteq E$, with $|M|$ maximal

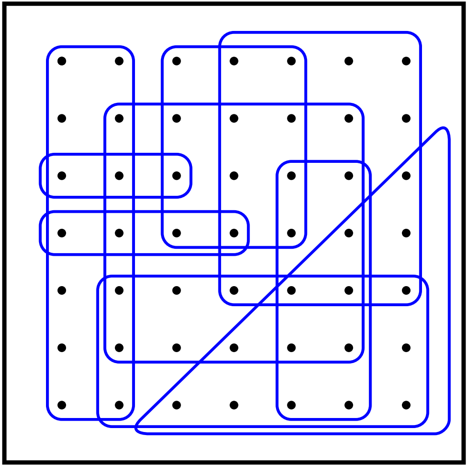
Maximum weight matching (MWM)

- Input: stream of weighted edges $(u, v, w_{uv}) \in [n] \times [n] \times \mathbb{R}^+$
- Goal: output matching $M \subseteq E$, with $w(M) = \sum_{e \in M} w(e)$ maximal

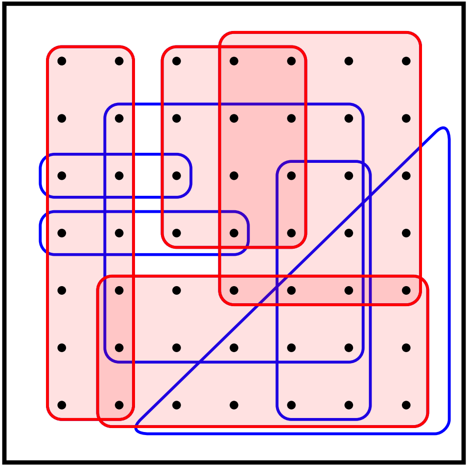
Maximum submodular-function matching (MSM) [Chakrabarti-Kale'14]

- Input: unweighted edges (u, v), plus *submodular* $f : 2^E \to \mathbb{R}^+$
- Goal: output matching $M \subseteq E$, with $f(M)$ maximal

Set Cover



Set Cover



Set Cover with Sets Streamed

- Input: stream of m sets, each $\subseteq [n]$
- Goal: cover universe $[n]$ using as few sets as possible

Set Cover with Sets Streamed

- Input: stream of m sets, each $\subseteq [n]$
- Goal: cover universe $[n]$ using as few sets as possible
  - Use sublinear (in m) space
  - Ideally $O(n \operatorname{polylog} n)$ ... "semi-streaming"
  - Need $\Omega(n \log n)$ space to *certify*: for each item, who covered it?

Think $m \geq n$

Road Map

- Results on Maximum Submodular Matching (MSM)
- Generalising MSM: constrained submodular maximisation

- Set Cover: upper bounds
- Set Cover: lower bounds, with proof outline

Maximum Submodular Matching

Input

- Stream of edges $\sigma = \langle e_1, e_2, \ldots, e_m \rangle$
- Valuation function $f : 2^E \to \mathbb{R}^+$
  - Submodular:
    $X \subseteq Y \subseteq E, e \in E \implies f(X + e) - f(X) \geq f(Y + e) - f(Y)$
  - Monotone:
    $X \subseteq Y \implies f(X) \leq f(Y)$
  - Normalised:
    $f(\varnothing) = 0$
- Oracle access to f: query at $X \subseteq E$, get $f(X)$
  - May only query at $X \subseteq$ (stream so far)

Goal

- Output matching $M \subseteq E$, with $f(M)$ ~~maximal~~ "large"
- Store $O(n)$ edges and f-values

Some Results on MSM

Can't solve MSM exactly

- MCM, approx $< e/(e-1) \implies$ space $\omega(n \operatorname{polylog} n)$ [Kapralov'13]
- Offline MSM, approx $< e/(e-1) \implies n^{\omega(1)}$ oracle calls
  - Via cardinality-constrained submodular max [Nemhauser-Wolsey'78]

Some Results on MSM

Can't solve MSM exactly

- MCM, approx $< e/(e-1) \implies$ space $\omega(n \operatorname{polylog} n)$ [Kapralov'13]
- Offline MSM, approx $< e/(e-1) \implies n^{\omega(1)}$ oracle calls
  - Via cardinality-constrained submodular max [Nemhauser-Wolsey'78]

Positive results, using $O(n)$ storage:

Theorem 1 MSM, one pass: 7.75-approx

Theorem 2 MSM, $(3+\varepsilon)$-approx in $O(e^{-3})$ passes

Some Results on MSM

Can't solve MSM exactly

- MCM, approx $< e/(e-1) \implies$ space $\omega(n \operatorname{polylog} n)$ [Kapralov'13]
- Offline MSM, approx $< e/(e-1) \implies n^{\omega(1)}$ oracle calls
  - Via cardinality-constrained submodular max [Nemhauser-Wolsey'78]

Positive results, using $O(n)$ storage:

Theorem 1 MSM, one pass: 7.75-approx

Theorem 2 MSM, $(3+\varepsilon)$-approx in $O(e^{-3})$ passes

More importantly:

Meta-Thm 1 Every *compliant* MWM approx alg $\rightarrow$ MSM approx alg

Some Results on MSM

Can't solve MSM exactly

- MCM, approx $< e/(e-1) \implies$ space $\omega(n \operatorname{polylog} n)$ [Kapralov'13]
- Offline MSM, approx $< e/(e-1) \implies n^{\omega(1)}$ oracle calls
  - Via cardinality-constrained submodular max [Nemhauser-Wolsey'78]

Positive results, using $O(n)$ storage:

Theorem 1 MSM, one pass: 7.75-approx

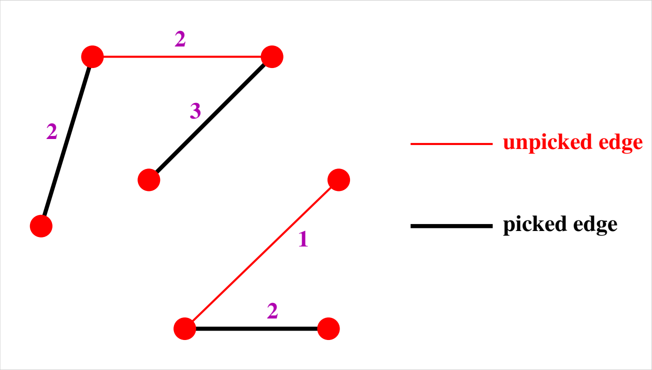
Theorem 2 MSM, $(3+\varepsilon)$-approx in $O(e^{-3})$ passes

More importantly:

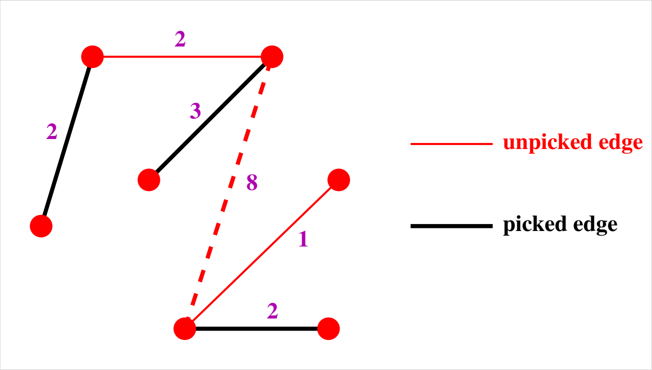
Meta-Thm 1 Every *compliant* MWM approx alg $\to$ MSM approx alg

Meta-Thm 2 Similarly, max weight *independent* set (MWIS) $\to$ MSIS

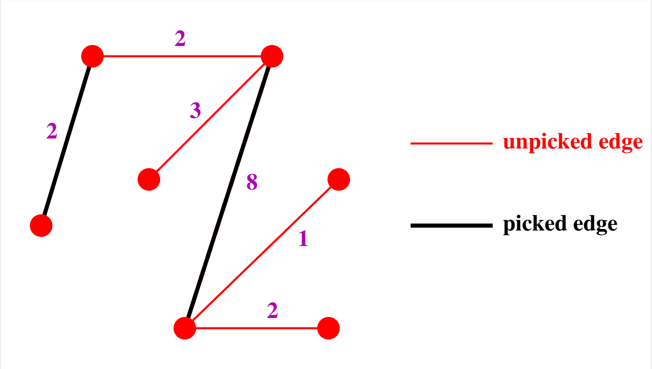
Compliant Algorithms for MWM



Compliant Algorithms for MWM



Compliant Algorithms for MWM



Maintain "current solution" M,
update if new edge improves it *sufficiently*

Compliant Algorithms for MWM: Details

Update of "current solution" M

- Given new edge e, pick "augmenting pair" (A, J)
  - $A \leftarrow \{e\}$
  - $J \leftarrow M \Cap A$ … edges in M that conflict with A
  - Ensure $w(A) \geq (1 + \gamma)w(J)$
- Update $M \leftarrow (M \setminus J) \cup A$

Choice of gain parameter

- $\gamma = 1$, approx factor 6 [Feigenbaum-K-M-S-Z'05]
- $\gamma = 1/\sqrt{2}$, approx factor 5.828 [McGregor'05]

Compliant Algorithms for MWM: Details

Update of "current solution" M

- Given new edge e, pick "augmenting pair" (A, J)
  - $\cancel{A \leftarrow \{e\}}$ $A \leftarrow$ "best" subset of 3-neighbourhood of e
  - $J \leftarrow M \Cap A$ ... edges in M that conflict with A
  - Ensure $w(A) \geq (1 + \gamma)w(J)$
- Update $M \leftarrow (M \setminus J) \cup A$

Choice of gain parameter

- $\gamma = 1$, approx factor 6 [Feigenbaum-K-M-S-Z'05]
- $\gamma = 1/\sqrt{2}$, approx factor 5.828 [McGregor'05]
- $\gamma = 1.717$, approx factor 5.585 [Zelke'08]

Compliant Algorithms for MWM: Details

Update of "current solution" M + pool of "shadow edges" S

- Given new edge e, pick "augmenting pair" (A, J)
  - $\cancel{A \leftarrow \{e\}}$ $A \leftarrow$ "best" subset of 3-neighbourhood of e
  - $J \leftarrow M \Cap A$ ... edges in M that conflict with A
  - Ensure $w(A) \geq (1+\gamma)w(J)$
- Update $M \leftarrow (M \setminus J) \cup A$
- Update $S \leftarrow$ appropriate subset of $(S \setminus A) \cup J$

Choice of gain parameter

- $\gamma = 1$, approx factor 6 [Feigenbaum-K-M-S-Z'05]
- $\gamma = 1/\sqrt{2}$, approx factor 5.828 [McGregor'05]
- $\gamma = 1.717$, approx factor 5.585 [Zelke'08]

Generic Compliant Algorithm and f-Extension for MSM

procedure $\textsc{Process-Edge}(e, M, S, \gamma)$

$(A, J) \leftarrow$ a well-chosen augmenting pair for M
with $A \subseteq M \cup S + e$, $w(A) \geq (1+\gamma)w(J)$

$M \leftarrow (M \setminus J) \cup A$

$S \leftarrow$ a well-chosen subset of $(S \setminus A) \cup J$

MWM alg $\mathcal{A}$ + submodular $f \rightarrow$ MSM alg $\mathcal{A}^f$ (the f-extension of $\mathcal{A}$)

Generic Compliant Algorithm and f-Extension for MSM

procedure PROCESS-EDGE(e, M, S, γ)
    $w(e) \leftarrow f(M \cup S + e) - f(M \cup S)$
    $(A, J) \leftarrow$ a well-chosen augmenting pair for M
        with $A \subseteq M \cup S + e$, $w(A) \geq (1 + \gamma)w(J)$
    $M \leftarrow (M \setminus J) \cup A$
    $S \leftarrow$ a well-chosen subset of $(S \setminus A) \cup J$

MWM alg $\mathcal{A}$ + submodular f $\rightarrow$ MSM alg $\mathcal{A}^f$ (the f-extension of $\mathcal{A}$)

Generic Compliant Algorithm and f-Extension for MSM

procedure $\textsc{Process-Edge}(e, M, S, \gamma)$
 $w(e) \leftarrow f(M \cup S + e) - f(M \cup S)$
 $(A, J) \leftarrow$ a well-chosen augmenting pair for M
  with $A \subseteq M \cup S + e$, $w(A) \geq (1 + \gamma)w(J)$
 $M \leftarrow (M \setminus J) \cup A$
 $S \leftarrow$ a well-chosen subset of $(S \setminus A) \cup J$

MWM alg $\mathcal{A}$ + submodular f $\rightarrow$ MSM alg $\mathcal{A}^f$ (the f-extension of $\mathcal{A}$)
MWIS (arbitrary ground set E, independent sets $\mathcal{I} \subseteq 2^E$) + f $\rightarrow$ MSIS

Generalise: Submodular Maximization (MWIS, MSIS)

procedure $\textsc{Process-Element}(e, I, S, \gamma)$

  $w(e) \leftarrow f(I \cup S + e) - f(I \cup S)$

  $(A, J) \leftarrow$ a well-chosen augmenting pair for I
  with $A \subseteq I \cup S + e$, $w(A) \geq (1 + \gamma)w(J)$

  $I \leftarrow (I \setminus J) \cup A$

  $S \leftarrow$ a well-chosen subset of $(S \setminus A) \cup J$

MWM alg $\mathcal{A}$ + submodular f $\rightarrow$ MSM alg $\mathcal{A}^f$ (the f-extension of $\mathcal{A}$)

MWIS (arbitrary ground set E, independent sets $\mathcal{I} \subseteq 2^E$) + f $\rightarrow$ MSIS

Further Applications: Hypermatchings

Stream of hyperedges $e_1, e_2, \ldots, e_m \subseteq [n]$, each $|e_i| \leq p$
Hypermatching = subset of pairwise disjoint edges

Further Applications: Hypermatchings

Stream of hyperedges $e_1, e_2, \ldots, e_m \subseteq [n]$, each $|e_i| \leq p$
Hypermatching = subset of pairwise disjoint edges

Multi-pass MSM algorithm (compliant)

- Augment using only current edge e
- Use $\gamma = 1$ for first pass, $\gamma = \varepsilon/(p+1)$ subsequently
- Make passes until solution doesn't improve much

Results

- $4p$-approx in one pass
- $(p + 1 + \varepsilon)$-approx in $O(\varepsilon^{-3})$ passes

Further Applications: Maximization Over Matroids

Stream of elements $e_1, e_2, \ldots, e_m$ from ground set E

Matroids $(E, \mathcal{I}_1), \ldots, (E, \mathcal{I}_p)$, given by *circuit oracles*:

$$\text{Given } A \subseteq E, \text{ returns } \begin{cases} ☺, & \text{if } A \in \mathcal{I}_i \\ \text{a circuit in } A, & \text{otherwise} \end{cases}$$

Independent sets, $\mathcal{I} = \bigcap_i \mathcal{I}_i$; size parameter $n = \max_{I \in \mathcal{I}} |I|$

Further Applications: Maximization Over Matroids

Stream of elements $e_1, e_2, \ldots, e_m$ from ground set E
Matroids $(E, \mathcal{I}_1), \ldots, (E, \mathcal{I}_p)$, given by *circuit oracles*:

$$\text{Given } A \subseteq E, \text{ returns } \begin{cases} ☺ , & \text{if } A \in \mathcal{I}_i \\ \text{a circuit in } A , & \text{otherwise} \end{cases}$$

Independent sets, $\mathcal{I} = \bigcap_i \mathcal{I}_i$; size parameter $n = \max_{I \in \mathcal{I}} |I|$
Recent MWIS algorithm (compliant) [Varadaraja'11]

- Augment using only current element e
- Remove $J = \{x_1, \ldots, x_p\}$,
  where $x_i :=$ lightest element in circuit formed in ith matroid

Further Applications: Maximization Over Matroids

Stream of elements $e_1, e_2, \ldots, e_m$ from ground set E
Independent sets, $\mathcal{I} = \bigcap_i \mathcal{I}_i$; size parameter $n = \max_{I \in \mathcal{I}} |I|$
Recent MWIS algorithm (compliant) [Varadaraja'11]

- Augment using only current element e
- Remove $J = \{x_1, \ldots, x_p\}$,
  where $x_i :=$ lightest element in circuit formed in ith matroid

Further Applications: Maximization Over Matroids

Stream of elements $e_1, e_2, \ldots, e_m$ from ground set E

Independent sets, $\mathcal{I} = \bigcap_i \mathcal{I}_i$; size parameter $n = \max_{I \in \mathcal{I}} |I|$

Recent MWIS algorithm (compliant) [Varadaraja'11]

- Augment using only current element e
- Remove $J = \{x_1, \ldots, x_p\}$,
  where $x_i :=$ lightest element in circuit formed in ith matroid

Follow paradigm: use f-extension of above algorithm

Results, using $O(n)$ storage

- $4p$-approx in one pass
- $(p + 1 + \varepsilon)$-approx in $O(\varepsilon^{-3})$ passes *

---

* Multi-pass analysis only works for *partition* matroids

Road Map

- Results on Maximum Submodular Matching (MSM) ✓
- Generalising MSM: constrained submodular maximisation ✓

- Set Cover: upper bounds
- Set Cover: lower bounds, with proof outline

Set Cover: Background

Offline results:

- Best possible poly-time approx $(1 \pm o(1)) \ln n$ [Johnson'74] [Slavík'96] [Lund-Yannakakis'94] [Dinur-Steurer'14]
- Simple greedy strategy gets $\ln n$-approx:
  - Repeatedly add set with highest *contribution*
  - Contribution := number of *new* elements covered

Streaming results:

- One pass semi-streaming $O(\sqrt{n})$-approx
- This is best possible in a single pass [Emek-Rosén'14]
- (More results in Indyk's talk)

Set Cover: Our Results

Upper bound

- With p passes, semi-streaming space, get $O(n^{1/(p+1)})$-approx
- Algorithm giving this approx based on very simple heuristic
- Deterministic

Lower bound

- Randomized
- In p passes, semi-streaming space, need $\Omega(n^{1/(p+1)}/p^2)$ space.
- Upper bound tight for all constant p
- Semi-streaming $O(\log n)$ approx requires $\Omega(\log n/\log\log n)$ passes

[Chakrabarti-Wirth'15]

Progressive Greedy Algorithm

procedure GreedyPass(stream σ, threshold τ, set Sol, array $Coverer$)
    for all (i, S) in σ **do**
        $C \leftarrow \{x : Coverer[x] \neq 0\}$ $\triangleright$ the already covered elements
        if $|S \setminus C| \geq \tau$ **then**
            $Sol \leftarrow Sol \cup \{i\}$
            for all $x \in S \setminus C$ **do** $Coverer[x] \leftarrow i$

procedure ProgGreedyNaive(stream σ, integer n, integer $p \geq 1$)
    $Coverer[1 \ldots n] \leftarrow 0^n$; $Sol \leftarrow \varnothing$
    for $j = 1$ **to** p **do** GreedyPass($\sigma, n^{1-j/p}, Sol, Coverer$)
    output $Sol, Coverer$

Progressive Greedy: Analysis Idea

Consider $p = 2$ passes

- First pass: admit sets iff contribution $\geq \sqrt{n}$
- Thus, first pass adds at most $\sqrt{n}$ sets to Sol

Progressive Greedy: Analysis Idea

Consider $p = 2$ passes

- First pass: admit sets iff contribution $\geq \sqrt{n}$
- Thus, first pass adds at most $\sqrt{n}$ sets to Sol
- Second pass: Opt cover remaining items with sets of contrib $\leq \sqrt{n}$
- Thus, Sol will cover the same using $\leq \sqrt{n}|Opt|$ sets

Progressive Greedy: Analysis Idea

Consider $p = 2$ passes

- First pass: admit sets iff contribution $\geq \sqrt{n}$
- Thus, first pass adds at most $\sqrt{n}$ sets to Sol
- Second pass: Opt cover remaining items with sets of contrib $\leq \sqrt{n}$
- Thus, Sol will cover the same using $\leq \sqrt{n}|Opt|$ sets

But wait, this uses *two* passes for $O(\sqrt{n})$ approx!

Progressive Greedy: Analysis Idea

Consider $p = 2$ passes

- First pass: admit sets iff contribution $\geq \sqrt{n}$
- Thus, first pass adds at most $\sqrt{n}$ sets to Sol
- Second pass: Opt cover remaining items with sets of contrib $\leq \sqrt{n}$
- Thus, Sol will cover the same using $\leq \sqrt{n}|Opt|$ sets

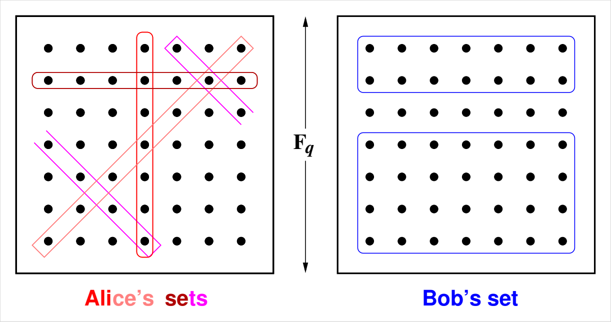
But wait, this uses *two* passes for $O(\sqrt{n})$ approx!

- Logic of last pass especially simple: add set if positive contrib
- Can *fold* this into previous one

Final result: p passes, $O(n^{1/(p+1)})$-approx

Lower Bound Idea: One Pass

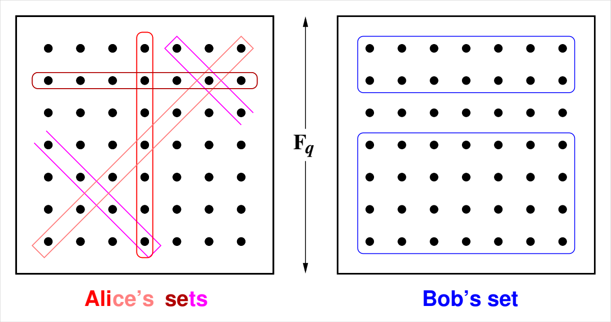
Reduce from INDEX: Alice gets $x \in \{0,1\}^n$, Bob gets $j \in [n]$, Alice talks to Bob, who must determine x_j. Requires $\Omega(n)$-bit message. [Ablayev'96]



$n = q^2$

Lower Bound Idea: One Pass

Reduce from INDEX: Alice gets $x \in \{0,1\}^n$, Bob gets $j \in [n]$, Alice talks to Bob, who must determine x_j. Requires $\Omega(n)$-bit message. [Ablayev'96]

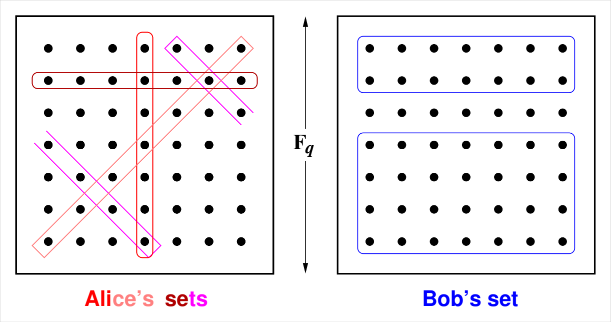


$n = q^2$

If Alice has Bob's *missing line*, then $|Opt| = 2$, else $|Opt| \geq q$

Lower Bound Idea: One Pass

Reduce from INDEX: Alice gets $x \in \{0,1\}^n$, Bob gets $j \in [n]$, Alice talks to Bob, who must determine x_j. Requires $\Omega(n)$-bit message. [Ablayev'96]



$n = q^2$

If Alice has Bob's *missing line*, then $|Opt| = 2$, else $|Opt| \geq q$

So $\Theta(\sqrt{n})$ approx requires $\Omega(\#\text{lines}) = \Omega(n)$ space

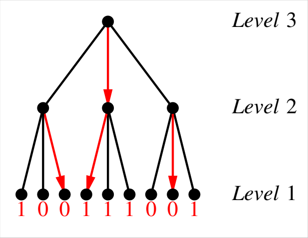
Tree Pointer Jumping

Multiplayer game $\text{TPJ}_{p+1,t}$ defined on complete $(p+1)$-level t-ary tree

- Pointer to child at each internal level-i node (known to Player i)
- Bit at each leaf node (known to Player 1)
- Goal: output (whp) bit reached by following pointers from root

Model: p rounds of communication

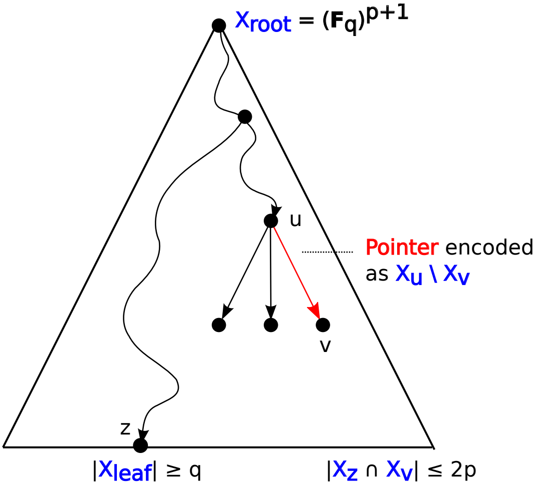
Each round: (Plr 1, Plr 2, . . . , Plr $(p+1)$)



Theorem: Longest message is $\Omega(t/p^2)$ bits [C.-Cormode-McGregor'08]

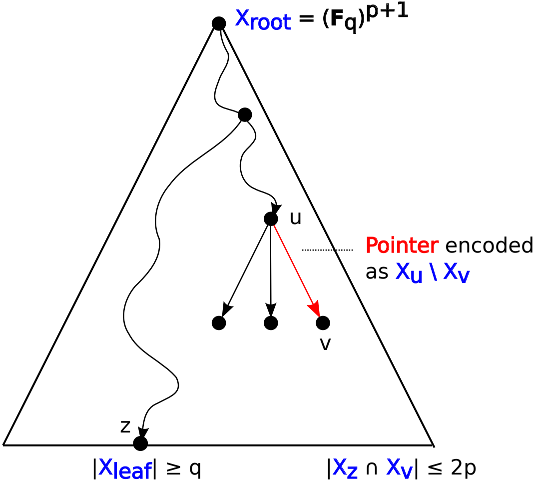
Edifices

Basic idea: Generalise affine plane to high-rank Buekenhout geometry



Edifices

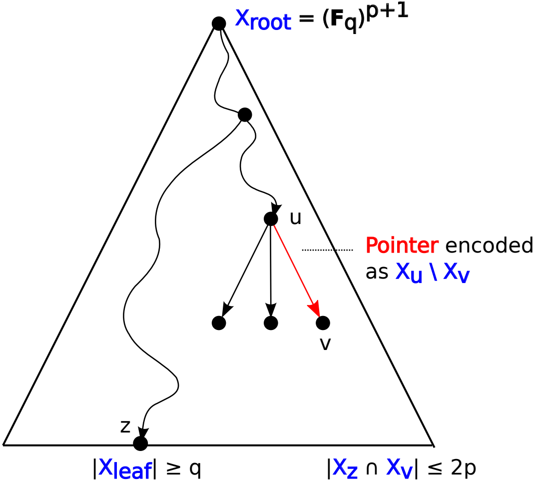
Basic idea: Generalise affine plane to high-rank Buekenhout geometry



- Universe $\mathbb{F}_q^{p+1}$
- Variety X_u at node u
- u above v
  $\Longrightarrow \; X_u \supseteq X_v$

Edifices

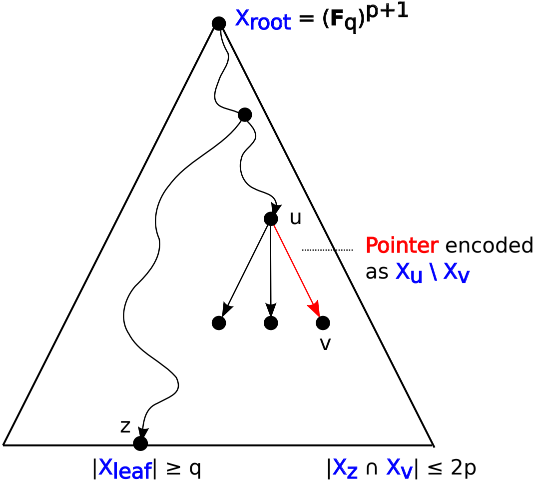
Basic idea: Generalise affine plane to high-rank Buekenhout geometry



- Universe $\mathbb{F}_q^{p+1}$
- Variety X_u at node u
- u above v
  $\implies X_u \supseteq X_v$
- Leaf z with bit $= 1$ encoded as set X_z

Edifices

Basic idea: Generalise affine plane to high-rank Buekenhout geometry



- Universe $\mathbb{F}_q^{p+1}$
- Variety X_u at node u
- u above v
  $\implies X_u \supseteq X_v$
- Leaf z with bit $= 1$ encoded as set X_z
- If player 1 has the *missing variety*, then $|Opt| = p + 1$, else $|Opt| \geq q/(2p)$

Construction of an Edifice

Basic idea: Varieties at leaves are low-degree curves, at level 2 they are low-degree surfaces, and so on.

Concern: Determining "cardinality" of algebraic variety over finite field is the stuff of difficult mathematics.

Construction of an Edifice

Basic idea: Varieties at leaves are low-degree curves, at level 2 they are low-degree surfaces, and so on.

Concern: Determining "cardinality" of algebraic variety over finite field is the stuff of difficult mathematics.

Our Solution: Define varieties using equations of special format

- Coordinates $(x, y_1, y_2, \ldots, y_p)$
- Equation at each edge of tree; at level i:

$$y_i = a_1 y_1 + \cdots a_{i-1} y_{i-1} + a_i f_{p+1-i}(x)$$
$$f_j(x) = \text{monic poly in } \mathbb{F}_q[x] \text{ of degree } p + j$$

- Variety X_u defined by equations on root-to-u path

Construction of an Edifice

Basic idea: Varieties at leaves are low-degree curves, at level 2 they are low-degree surfaces, and so on.

Concern: Determining "cardinality" of algebraic variety over finite field is the stuff of difficult mathematics.

Our Solution: Define varieties using equations of special format

- Coordinates $(x, y_1, y_2, \ldots, y_p)$
- Equation at each edge of tree; at level i:

$$y_i = a_1 y_1 + \cdots a_{i-1} y_{i-1} + a_i f_{p+1-i}(x)$$
$$f_j(x) = \text{monic poly in } \mathbb{F}_q[x] \text{ of degree } p + j$$

Cardinality bound via much simpler mathematics.

- Schwartz-Zippel lemma
- Linear independence arguments via row reduction

Final Remarks

Combinatorial optimisation: old topic, but relatively new territory for data stream algorithms

- Potential for many new research questions
- Stronger or more general results on submodular maximization? Some new work in [Chekuri-Gupta-Quanrud'15]
- Lower bounds for submodular maximization?
- Fuller understanding of possible tradeoff for set cover?