Composable Core-sets for Determinant Maximization: A Simple Near-Optimal Algorithm

Piotr Indyk
MIT

Sepideh Mahabadi
TTIC

Shayan Oveis Gharan
U. of Washington

Alireza Rezaei
U. of Washington
Volume (Determinant) Maximization Problem

**Input:** a set of $n$ vectors $V \in \mathbb{R}^d$ and a parameter $k \leq d$, $k = 2$.
Volume (Determinant) Maximization Problem

**Input:** a set of $n$ vectors $V \in \mathbb{R}^d$ and a parameter $k \leq d$,

**Output:** a subset $S \subset V$ of size $k$ with the maximum volume

- Parallelepiped spanned by the points in $S$
Volume (Determinant) Maximization Problem

**Input:** a set of $n$ vectors $V \in \mathbb{R}^d$ and a parameter $k \leq d$,

**Output:** a subset $S \subset V$ of size $k$ with the maximum volume

- Parallelepiped spanned by the points in $S$
Determinant (Volume) Maximization Problem

**Input:** a set of $n$ vectors $V \in \mathbb{R}^d$ and a parameter $k \leq d$,

**Output:** a subset $S \subset V$ of size $k$ with the maximum volume

- Parallelepiped spanned by the points in $S$

$$
\begin{bmatrix}
v_1 & v_2 & \ldots & v_n
\end{bmatrix}
$$

**Equivalent Formulation:**

Reuse $V$ to denote the matrix where its columns are the vectors in $V$
Determinant (Volume) Maximization Problem

**Input:** a set of \( n \) vectors \( V \in \mathbb{R}^d \) and a parameter \( k \leq d \),

**Output:** a subset \( S \subset V \) of size \( k \) with the maximum volume

- Parallelepiped spanned by the points in \( S \)

\[
\begin{bmatrix}
v_1 \\
v_2 \\
\vdots \\
v_n
\end{bmatrix} \times
\begin{bmatrix}
v_1 & v_2 & \cdots & v_n
\end{bmatrix} =
\begin{bmatrix}
i \\
j
\end{bmatrix}
\]

\[
M_{i,j} = v_i \cdot v_j
\]

**Equivalent Formulation:**

Reuse \( V \) to denote the matrix where its columns are the vectors in \( V \)

- Let \( M \) be the gram matrix \( V^T V \)
**Determinant (Volume) Maximization Problem**

**Input:** a set of $n$ vectors $V \in \mathbb{R}^d$ and a parameter $k \leq d$,

**Output:** a subset $S \subset V$ of size $k$ with the maximum volume

- Parallelepiped spanned by the points in $S$

---

**Equivalent Formulation:**

Reuse $V$ to denote the matrix where its columns are the vectors in $V$

- Let $M$ be the gram matrix $V^T V$
- Choose $S$ such that $\det(M_{S,S})$ is maximized

$$M_{i,j} = v_i \cdot v_j$$

$$\det(M_{S,S}) = Vol(S)^2$$
What is known?

- Hard to approximate within a factor of $2^{ck}$ [CMI’13]
What is known?

• Hard to approximate within a factor of $2^{ck}$ [CMI’13]
• Best algorithm: $e^k$-approximation [Nik’15]
What is known?

- Hard to approximate within a factor of $2^{ck}$ [CMI’13]
- Best algorithm: $e^k$-approximation [Nik’15]
- **Greedy** is a popular algorithm: achieves approximation factor $k$!
  - $U \leftarrow \emptyset$
  - For $k$ iterations,
    - Add to $U$ the farthest point from the subspace spanned by $U$
What is known?

• Hard to approximate within a factor of $2^{ck}$ [CMI’13]
• Best algorithm: $e^k$-approximation [Nik’15]
• **Greedy** is a popular algorithm: achieves approximation factor $k$!
  
  - $U \leftarrow \emptyset$
  - For $k$ iterations,
    - Add to $U$ the farthest point from the subspace spanned by $U$

![Diagram](image-url)
What is known?

- Hard to approximate within a factor of $2^{ck}$ [CMI’13]
- Best algorithm: $e^k$-approximation [Nik’15]
- **Greedy** is a popular algorithm: achieves approximation factor $k$!
  - $U \leftarrow \emptyset$
  - For $k$ iterations,
    - Add to $U$ the farthest point from the subspace spanned by $U$
What is known?

- Hard to approximate within a factor of $2^{ck}$ [CMI’13]
- Best algorithm: $e^k$-approximation [Nik’15]
- **Greedy** is a popular algorithm: achieves approximation factor $k$!
  - $U \leftarrow \emptyset$
  - For $k$ iterations,
    - Add to $U$ the farthest point from the subspace spanned by $U$

![Diagram](image)
What is known?

- Hard to approximate within a factor of $2^{ck}$ [CMI’13]
- Best algorithm: $e^k$-approximation [Nik’15]
- **Greedy** is a popular algorithm: achieves approximation factor $k$!
  - $U \leftarrow \emptyset$
  - For $k$ iterations,
    - Add to $U$ the farthest point from the subspace spanned by $U$
What is known?

• Hard to approximate within a factor of $2^{ck}$ [CMI’13]
• Best algorithm: $e^k$-approximation [Nik’15]
• **Greedy** is a popular algorithm: achieves approximation factor $k!
  - U \leftarrow \emptyset
  - For $k$ iterations,
    - Add to $U$ the farthest point from the subspace spanned by $U$

• Greedy performs very well in practice

$k = 2$
Determinantal Point Processes (DPP)

**DPP:** Very popular probabilistic model, where given a set of vectors $V$, samples any $k$-subset $S$ with probability proportional to this determinant.

- Maximum a posteriori (MAP) decoding is determinant maximization
- Volume/determinant is a notion of *diversity*

**References:**
- NeurIPS’18 Tutorial, *Negative Dependence, Stable Polynomials, and All That*, Jegelka, Sra
Application: Diversity Maximization

Given a set of objects, how to pick a few of them while maximizing diversity?
Application: Diversity Maximization

Given a set of objects, how to pick a few of them while maximizing diversity?

• Searching
Application: Diversity Maximization

Given a set of objects, how to pick a few of them while maximizing diversity?

• Searching
Application: Diversity Maximization

Objects (documents, images, etc) → Feature Vectors → Points in a high dimensional space
Application: Diversity Maximization

**Input:** a set of $n$ vectors $V \subset \mathbb{R}^d$ and a parameter $k$,

**Goal:** pick $k$ points while maximizing “diversity”.

$k = 3$
Determinantal Point Processes (DPP)

**DPP:** Very popular probabilistic model, where given a set of vectors $V$, samples any $k$-subset $S$ with probability proportional to this determinant.

- Maximum a posteriori (MAP) decoding is determinant maximization
- Volume/determinant is a notion of diversity

**Applications**

-[MJK'17, GCGS'14] Video summarization
-[KT+'12, CGGS'15, KT'11] Document summarization
-[YFZ+'16] Tweet generation
-[LCYO'16] Object detection
-....
Determinantal Point Processes (DPP)

**DPP:** Very popular probabilistic model, where given a set of vectors $V$, samples any $k$-subset $S$ with probability proportional to this determinant.

- Maximum a posteriori (MAP) decoding is determinant maximization
- Volume/determinant is a notion of diversity

Applications

- Video summarization
- Document summarization
- Tweet generation
- Object detection

Most applications deal with **massive data**
- Lots of effort for solving the problem in massive data models of computation
- e.g. streaming, distributed, parallel
Determinantal Point Processes (DPP)

**DPP:** Very popular probabilistic model, where given a set of vectors $V$, samples any $k$-subset $S$ with probability proportional to this determinant.

- Maximum a posteriori (MAP) decoding is determinant maximization
- Volume/determinant is a notion of *diversity*

---

**Applications**

- Video summarization: [MJK’17, GCGS’14]
- Document summarization: [KT+’12, CGGS’15, KT’11]
- Tweet generation: [YFZ+’16]
- Object detection: [LCYO’16]

**Composable Core-sets**

- Most applications deal with *massive data*
- Lots of effort for solving the problem in massive data models of computation [MJK’17, WIB’14, PJG+’14, MKSK’13, MKBK’15, MZ’15, MZ’15, BENW’15]
- *e.g. streaming, distributed, parallel*
Core-sets

Core-sets [AHV’05]: a subset $U$ of the data $V$ that represents it well

Solving the problem over $U$ gives a good approximation of solving the problem over $V$
Composable Core-sets

**Core-sets [AHV’05]:** a subset $U$ of the data $V$ that represents it well

**Composable Core-sets [AAIMV’13 and IMM’14]:**

The union of coresets is a coreset for the union
**Composable Core-sets**

**Core-sets [AHV’05]:** a subset $U$ of the data $V$ that represents it well

**Composable Core-sets [AAIMV’13 and IMM’14]:**

The union of coresets is a coreset for the union

- Let $f$ be an optimization function
  - E.g. $f(V)$: solution to $k$ determinant maximization
Composable Core-sets

**Core-sets [AHV’05]:** a subset $U$ of the data $V$ that represents it well

**Composable Core-sets [AAIMV’13 and IMMM’14]:**

The union of coresets is a coreset for the union

- Let $f$ be an optimization function
  - E.g. $f(V)$: solution to $k$ determinant maximization
- Multiple data sets $V_1, \cdots, V_m$ and their coresets $U_1 \subset V_1, \cdots, U_m \subset V_m$,
  - $f(U_1 \cup \cdots \cup U_m)$ approximates $f(V_1 \cup \cdots \cup V_m)$ by a factor $\alpha$
Composable Core-sets

Core-sets [AHV’05]: a subset $U$ of the data $V$ that represents it well

Composable Core-sets [AAIMV’13 and IMMM’14]:
The union of coresets is a coreset for the union

- Let $f$ be an optimization function
  - E.g. $f(V)$: solution to $k$ determinant maximization
- Multiple data sets $V_1, \ldots, V_m$ and their coresets $U_1 \subset V_1, \ldots, U_m \subset V_m$,
  - $f(U_1 \cup \cdots \cup U_m)$ approximates $f(V_1 \cup \cdots \cup V_m)$ by a factor $\alpha$
Composable Core-sets

Core-sets [AHV’05]: a subset $U$ of the data $V$ that represents it well

Composable Core-sets [AAIMV’13 and IMM’14]:

The union of coresets is a coreset for the union

- Let $f$ be an optimization function
  - E.g. $f(V)$: solution to $k$ determinant maximization
- Multiple data sets $V_1, \ldots, V_m$ and their coresets $U_1 \subset V_1, \ldots, U_m \subset V_m,$
  - $f(U_1 \cup \cdots \cup U_m)$ approximates $f(V_1 \cup \cdots \cup V_m)$ by a factor $\alpha$
Composable Core-sets

**Core-sets [AHV’05]:** a subset $U$ of the data $V$ that represents it well

**Composable Core-sets [AAIMV’13 and IMM’14]:**

The union of coresets is a coreset for the union

- Let $f$ be an optimization function
  - E.g. $f(V)$: solution to $k$ determinant maximization
- Multiple data sets $V_1, \ldots, V_m$ and their coresets $U_1 \subset V_1, \ldots, U_m \subset V_m$,
  - $f(U_1 \cup \cdots \cup U_m)$ approximates $f(V_1 \cup \cdots \cup V_m)$ by a factor $\alpha$
Composable Core-sets

**Core-sets** [AHV’05]: a subset $U$ of the data $V$ that represents it well

**Composable Core-sets** [AAIMV’13 and IMM’14]:
The union of coresets is a coreset for the union

- Let $f$ be an optimization function
  - E.g. $f(V)$: solution to $k$ determinant maximization
- Multiple data sets $V_1, \ldots, V_m$ and their coresets $U_1 \subseteq V_1, \ldots, U_m \subseteq V_m$,
  - $f(U_1 \cup \cdots \cup U_m)$ approximates $f(V_1 \cup \cdots \cup V_m)$ by a factor $\alpha$
Composable Core-sets

**Core-sets [AHV’05]:** a subset $U$ of the data $V$ that represents it well

**Composable Core-sets [AAIMMV’13 and IMMM’14]:**
The union of coresets is a coreset for the union

- Let $f$ be an optimization function
  - E.g. $f(V)$: solution to $k$ determinant maximization
- Multiple data sets $V_1, \ldots, V_m$ and their coresets $U_1 \subset V_1, \ldots, U_m \subset V_m$,
  - $f(U_1 \cup \ldots \cup U_m)$ approximates $f(V_1 \cup \ldots \cup V_m)$ by a factor $\alpha$

✓ Composable Core-sets have been studied for the **diversity Maximization** problems, for other notions of diversity: minimum pairwise distance, sum of pairwise distances, etc.

✓ Determinant maximization is a “higher order” notion of diversity
Applications: Streaming Computation

- **Streaming Computation:**
  - Processing sequence of \( n \) data elements “on the fly”
  - limited Storage
Applications: Streaming Computation

• **Streaming Computation:**
  • Processing sequence of $n$ data elements “on the fly”
  • limited Storage

• **Composable Core-set**
  • Divide into chunks
Applications: Streaming Computation

- **Streaming Computation:**
  - Processing sequence of $n$ data elements “on the fly”
  - limited Storage

- **Composable Core-set**
  - Divide into chunks
  - Compute Core-set for each chunk as it arrives
Applications: Streaming Computation

• **Streaming Computation:**
  • Processing sequence of \( n \) data elements “on the fly”
  • limited Storage

• **Composable Core-set**
  • Divide into chunks
  • Compute Core-set for each chunk as it arrives
Applications: Streaming Computation

- **Streaming Computation:**
  - Processing sequence of $n$ data elements “on the fly”
  - limited Storage

- **Composable Core-set**
  - Divide into chunks
  - Compute Core-set for each chunk as it arrives
  - Space goes down from $n$ to $\sqrt{n}$
Applications: Distributed Computation

• **Streaming Computation**

• **Distributed System:**
  • Each machine holds a block of data.
  • A composable core-set is computed and sent to the server

![Diagram showing the process of streaming computation and distributed system components like Mapper and Reducer.](image)
Applications: Improving Runtime

• Streaming Computation
• Distributed System
• Similar framework for improving the runtime
Can we get a composable core-set of small size for the determinant maximization problem?
Composable Core-sets for Volume Maximization

<table>
<thead>
<tr>
<th></th>
<th>[IMOR’18]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Approximation</strong></td>
<td>$\tilde{O}(k)^{k/2}$</td>
</tr>
<tr>
<td><strong>Core-set Size</strong></td>
<td>$\tilde{O}(k)$</td>
</tr>
<tr>
<td><strong>Simple?</strong></td>
<td>×</td>
</tr>
</tbody>
</table>

**LP-based Optimal Approximation Algorithm of [IMOR’18]:**

There exists a polynomial time algorithm for computing an $\tilde{O}(k)^{k/2}$-composable core-set of size $\tilde{O}(k)$ for the volume maximization problem.
## Composable Core-sets for Volume Maximization

<table>
<thead>
<tr>
<th></th>
<th>Lower Bound</th>
<th>[IMOR’18]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Approximation</strong></td>
<td>$\Omega(k)^{\frac{k}{2}o(k)}$</td>
<td>$\widetilde{O}(k)^{\frac{k}{2}}$</td>
</tr>
<tr>
<td><strong>Core-set Size</strong></td>
<td>$k^{O(1)}$</td>
<td>$\widetilde{O}(k)$</td>
</tr>
<tr>
<td><strong>Simple?</strong></td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>

**Lower bound [IMOR’18]:**

Any composable core-set of size $k^{O(1)}$ for the volume maximization problem must have an approximation factor of $\Omega(k)^{\frac{k}{2}(1-o(1))}$. 
The widely used Greedy algorithm produces a composable core-set of size \( k \) with approximation factor \( O(C^{k^2}) \).
Our Results

<table>
<thead>
<tr>
<th></th>
<th>Lower Bound</th>
<th>[IMOR’18]</th>
<th>Greedy</th>
<th>Local Search</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approximation</td>
<td>( \Omega(k) \frac{k^{\frac{1}{2}}}{2^{o(k)}} )</td>
<td>( \tilde{O}(k)^{\frac{k}{2}} )</td>
<td>( O(C^{k^2}) )</td>
<td>( O(k^k) )</td>
</tr>
<tr>
<td>Core-set Size</td>
<td>( k^{O(1)} )</td>
<td>( \tilde{O}(k) )</td>
<td>( k )</td>
<td>( k )</td>
</tr>
<tr>
<td>Simple?</td>
<td>( \times )</td>
<td>( \sqrt{\ } )</td>
<td>( \sqrt{\ } )</td>
<td>( \sqrt{\ } )</td>
</tr>
</tbody>
</table>

The Local Search Algorithm produces a composable core-set of size \( k \) with approximation factor \( O(k)^{2k} \).
This Talk

The Local Search Algorithm produces a composable core-set of size $k$ with approximation factor $O(k)^k$ for the volume maximization problem.
This Talk

The Local Search Algorithm produces a composable core-set of size $k$ with approximation factor $O(k)^k$ for the volume maximization problem.

In comparison to the optimal core-set algorithm

- Approximation $O(k)^k$ as opposed to $O(k \log k)^{k/2}$
- Smaller Size $k$ as opposed to $O(k \log k)$
- Simpler to implement (similar to Greedy)
- Better performance in practice
The Local Search Algorithm

**Input:** a set $V$ of $n$ points and a parameter $k$

1. Start with an arbitrary subset of $k$ points $S \subseteq V$

2. While there exists a point $p \in V \setminus S$ and $q \in S$ s.t. replacing $q$ with $p$ increases the volume, then swap them, i.e., $S = S \cup \{p\} \setminus \{q\}$
The Local Search Algorithm

**Input:** a set $V$ of $n$ points and a parameter $k$

1. Start with an arbitrary subset of $k$ points $S \subseteq V$

2. While there exists a point $p \in V \setminus S$ and $q \in S$ s.t. replacing $q$ with $p$ increases the volume, then swap them, i.e., $S = S \cup \{p\} \setminus \{q\}$
The Local Search Algorithm

**Input:** a set $V$ of $n$ points and a parameter $k$

1. Start with an arbitrary subset of $k$ points $S \subseteq V$

2. While there exists a point $p \in V \setminus S$ and $q \in S$ s.t. replacing $q$ with $p$ increases the volume, then swap them, i.e., $S = S \cup \{p\} \setminus \{q\}$
The Local Search Algorithm

**Input:** a set $V$ of $n$ points and a parameter $k$

1. Start with an arbitrary subset of $k$ points $S \subseteq V$

2. While there exists a point $p \in V \setminus S$ and $q \in S$ s.t. replacing $q$ with $p$ increases the volume, then swap them, i.e., $S = S \cup \{p\} \setminus \{q\}$
The Local Search Algorithm

Input: a set $V$ of $n$ points and a parameter $k$

1. Start with an arbitrary subset of $k$ points $S \subseteq V$

2. While there exists a point $p \in V \setminus S$ and $q \in S$ s.t. replacing $q$ with $p$ increases the volume, then swap them, i.e., $S = S \cup \{p\} \setminus \{q\}$
The Local Search Algorithm

**Input:** a set $V$ of $n$ points and a parameter $k$

1. Start with an arbitrary subset of $k$ points $S \subseteq V$

2. While there exists a point $p \in V \setminus S$ and $q \in S$ s.t. replacing $q$ with $p$ increases the volume, then swap them, i.e., $S = S \cup \{p\} \setminus \{q\}$
To bound the run time

**Input:** a set $V$ of $n$ points and a parameter $k$

1. Start with an *arbitrary* subset of $k$ points $S \subseteq V$

2. While there exists a point $p \in V \setminus S$ and $q \in S$ s.t. replacing $q$ with $p$ increases the volume, then swap them, i.e., $S = S \cup \{p\} \setminus \{q\}$

Start with a crude approximation (Greedy algorithm)

If it increases by at least a factor of $(1 + \epsilon)$
Checking the condition

**Input:** a set $V$ of $n$ points and a parameter $k$

1. Start with an arbitrary subset of $k$ points $S \subseteq V$

2. While there exists a point $p \in V \setminus S$ and $q \in S$ s.t. replacing $q$ with $p$ *increases the volume*, then swap them, i.e., $S = S \cup \{p\} \setminus \{q\}$

$$\text{dist}(p, H_{S \setminus \{q\}}) > \text{dist}(q, H_{S \setminus \{q\}})$$

$(k - 1)$-dimensional Subspace
Main Lemma [informal]:
Local Search preserves maximum distance to “all” subspaces of dimension $k - 1$
Main Lemma [informal]:
Local Search preserves maximum distance to “all” subspaces of dimension $k - 1$.

- $V$ is the point set
- $S = LS(V)$ is the core-set produced by local search
Main Lemma [informal]:
Local Search preserves maximum distance to “all” subspaces of dimension $k - 1$

- $V$ is the point set
- $S = LS(V)$ is the core-set produced by local search

Main Lemma [formal]:
For any $(k - 1)$-dimensional subspace $G$, the maximum distance of the point set to $G$ is approximately preserved

$$\max_{q \in S} \text{dist}(q, G) \geq \frac{1}{2k} \cdot \max_{p \in V} \text{dist}(p, G)$$
Main Lemma [formal]:
For any \((k - 1)\)-dimensional subspace \(G\), the maximum distance of the point set to \(G\) is approximately preserved

\[
\max_{s \in S} dist(q, G) \geq \frac{1}{2k} \cdot \max_{p \in V} dist(p, G)
\]

• Let \(p \in V\) be a point
Main Lemma [formal]:
For any \((k - 1)\)-dimensional subspace \(G\), the maximum distance of the point set to \(G\) is approximately preserved

\[
\max_{s \in S} \text{dist}(q, G) \geq \frac{1}{2k} \cdot \max_{p \in V} \text{dist}(p, G)
\]

• Let \(p \in V\) be a point
• Let \(G\) be a \((k - 1)\)-dimensional subspace.
Main Lemma [formal]:
For any \((k - 1)\)-dimensional subspace \(G\), the maximum distance of the point set to \(G\) is approximately preserved

\[
\max_{s \in S} \text{dist}(q, G) \geq \frac{1}{2k} \cdot \max_{p \in V} \text{dist}(p, G)
\]

• Let \(p \in V\) be a point
• Let \(G\) be a \((k - 1)\)-dimensional subspace.
• Assume for any \(q \in S\), \(d(q, G) \leq x\)
Main Lemma [formal]:
For any \((k - 1)\)-dimensional subspace \(G\), the maximum distance of the point set to \(G\) is approximately preserved

\[
\max_{s \in S} \text{dist}(q, G) \geq \frac{1}{2k} \cdot \max_{p \in V} \text{dist}(p, G)
\]

• Let \(p \in V\) be a point
• Let \(G\) be a \((k - 1)\)-dimensional subspace.
• Assume for any \(q \in S\), \(d(q, G) \leq x\)
Main Lemma [formal]:
For any \((k - 1)\)-dimensional subspace \(G\), the maximum distance of the point set to \(G\) is approximately preserved

\[
\max_{s \in S} \text{dist}(q, G) \geq \frac{1}{2k} \cdot \max_{p \in V} \text{dist}(p, G)
\]

- Let \(p \in V\) be a point
- Let \(G\) be a \((k - 1)\)-dimensional subspace.
- Assume for any \(q \in S\), \(d(q, G) \leq x\)

Lemma: \(d(p, G) \leq 2kx\)
Let $V = \bigcup_i V_i$ be the union of the point sets.

**Main Theorem**

Local Search produces a core-set for volume maximization
Main Theorem

Local Search produces a core-set for volume maximization

Let $V = \bigcup_i V_i$ be the union of the point sets

Let $S = \bigcup_i S_i$ be the union of core-sets
Main Theorem
Local Search produces a core-set for volume maximization

Let \( V = \bigcup_i V_i \) be the union of the point sets

Let \( S = \bigcup_i S_i \) be the union of core-sets

Let \( \text{Opt} = \{o_1, ..., o_k\} \subset V \) be the optimal subset of points maximizing the volume
Let \( V = \bigcup_i V_i \) be the union of the point sets.

Let \( S = \bigcup_i S_i \) be the union of core-sets.

Let \( \text{Opt} = \{o_1, \ldots, o_k\} \subset V \) be the optimal subset of points maximizing the volume.

**Main Theorem**

Local Search produces a core-set for volume maximization

\[
\begin{align*}
\text{Sol} &\leftarrow \text{Opt} \\
\text{For } i = 1 \text{ to } k \\
&\quad \text{Let } q_i \in S \text{ be the point that is farthest away from } H_{\text{Sol}\{o_i\}} \\
&\quad \text{Sol } \leftarrow \text{Sol } \cup \{q_i\} \setminus \{o_i\}
\end{align*}
\]
Main Theorem
Local Search produces a core-set for volume maximization

Let $V = \bigcup_i V_i$ be the union of the point sets

Let $S = \bigcup_i S_i$ be the union of core-sets

Let $Opt = \{o_1, \ldots, o_k\} \subset V$ be the optimal subset of points maximizing the volume

Sol $\leftarrow$ Opt
For $i = 1$ to $k$
  • Let $q_i \in S$ be the point that is farthest away from $H_{Sol \setminus \{o_i\}}$
  • Sol $\leftarrow$ Sol $\cup \{q_i\} \setminus \{o_i\}$
Main Theorem
Local Search produces a core-set for volume maximization

Let \( V = \bigcup_i V_i \) be the union of the point sets

Let \( S = \bigcup_i S_i \) be the union of core-sets

Let \( \text{Opt} = \{ o_1, \ldots, o_k \} \subset V \) be the optimal subset of points maximizing the volume

\[ \text{Sol} \leftarrow \text{Opt} \]

For \( i = 1 \) to \( k \)

- Let \( q_i \in S \) be the point that is farthest away from \( H_{\text{Sol} \setminus \{ o_i \}} \)
- \( \text{Sol} \leftarrow \text{Sol} \cup \{ q_i \} \setminus \{ o_i \} \)
Main Theorem
Local Search produces a core-set for volume maximization

Let \( V = \bigcup_i V_i \) be the union of the point sets

Let \( S = \bigcup_i S_i \) be the union of core-sets

Let \( Opt = \{ o_1, \ldots, o_k \} \subset V \) be the optimal subset of points maximizing the volume

\[
\begin{align*}
Sol & \leftarrow Opt \\
\text{For } i = 1 \text{ to } k \\
& \quad \text{• Let } q_i \in S \text{ be the point that is farthest away from } H_{Sol\setminus\{o_i\}} \\
& \quad \text{• } Sol \leftarrow Sol \cup \{q_i\} \setminus \{o_i\}
\end{align*}
\]
**Main Theorem**

Local Search produces a core-set for volume maximization

Let $V = \bigcup_i V_i$ be the union of the point sets

Let $S = \bigcup_i S_i$ be the union of core-sets

Let $Opt = \{o_1, \ldots, o_k\} \subset V$ be the optimal subset of points maximizing the volume

---

$Sol \leftarrow Opt$

For $i = 1$ to $k$

- Let $q_i \in S$ be the point that is farthest away from $H_{Sol \setminus \{o_i\}}$
- $Sol \leftarrow Sol \cup \{q_i\} \setminus \{o_i\}$

---
Main Theorem
Local Search produces a core-set for volume maximization

Let $V = \bigcup_i V_i$ be the union of the point sets

Let $S = \bigcup_i S_i$ be the union of core-sets

Let $Opt = \{o_1, ..., o_k\} \subset V$ be the optimal subset of points maximizing the volume

$Sol \leftarrow Opt$
For $i = 1$ to $k$
• Let $q_i \in S$ be the point that is farthest away from $H_{Sol \setminus \{o_i\}}$
• $Sol \leftarrow Sol \cup \{q_i\} \setminus \{o_i\}$
Main Theorem
Local Search produces a core-set for volume maximization

Let $V = \bigcup_i V_i$ be the union of the point sets

Let $S = \bigcup_i S_i$ be the union of core-sets

Let $Opt = \{o_1, ..., o_k\} \subset V$ be the optimal subset of points maximizing the volume

\[
Sol \leftarrow Opt
\]

For $i = 1$ to $k$

• Let $q_i \in S$ be the point that is farthest away from $H_{Sol\setminus\{o_i\}}$

• $Sol \leftarrow Sol \cup \{q_i\} \setminus \{o_i\}$
Main Theorem
Local Search produces a core-set for volume maximization

Let $V = \bigcup_i V_i$ be the union of the point sets

Let $S = \bigcup_i S_i$ be the union of core-sets

Let $Opt = \{o_1, \ldots, o_k\} \subseteq V$ be the optimal subset of points maximizing the volume

$Sol \leftarrow Opt$

For $i = 1$ to $k$
- Let $q_i \in S$ be the point that is farthest away from $H_{Sol\setminus\{o_i\}}$
- $Sol \leftarrow Sol \cup \{q_i\} \setminus \{o_i\}$
Main Theorem
Local Search produces a core-set for volume maximization

Let \( V = \bigcup_i V_i \) be the union of the point sets

Let \( S = \bigcup_i S_i \) be the union of core-sets

Let \( Opt = \{o_1, \ldots, o_k\} \subset V \) be the optimal subset of points maximizing the volume

\[
Sol \leftarrow Opt \\
For \ i = 1 \ to \ k \\
\quad \bullet \ Let \ q_i \in S \ be \ the \ point \ that \ is \ farthest \ away \ from \ H_{Sol\backslash\{o_i\}} \\
\quad \bullet \ Sol \leftarrow Sol \cup \{q_i\} \setminus \{o_i\}
\]
Main Theorem
Local Search produces a core-set for volume maximization

Let $V = \bigcup_i V_i$ be the union of the point sets

Let $S = \bigcup_i S_i$ be the union of core-sets

Let $Opt = \{o_1, \ldots, o_k\} \subset V$ be the optimal subset of points maximizing the volume

$$Sol \leftarrow Opt$$

For $i = 1$ to $k$

• Let $q_i \in S$ be the point that is farthest away from $H_{Sol \setminus \{o_i\}}$

• $Sol \leftarrow Sol \cup \{q_i\} \setminus \{o_i\}$
Let $V = \bigcup_i V_i$ be the union of the point sets

Let $S = \bigcup_i S_i$ be the union of core-sets

Let $Opt = \{o_1, \ldots, o_k\} \subset V$ be the optimal subset of points maximizing the volume

$Sol \leftarrow Opt$

For $i = 1$ to $k$

- Let $q_i \in S$ be the point that is farthest away from $H_{Sol\setminus \{o_i\}}$
- $Sol \leftarrow Sol \cup \{q_i\} \setminus \{o_i\}$
Main Theorem
Local Search produces a core-set for volume maximization

Let $V = \bigcup_i V_i$ be the union of the point sets

Let $S = \bigcup_i S_i$ be the union of core-sets

Let $Opt = \{o_1, \ldots, o_k\} \subset V$ be the optimal subset of points maximizing the volume

$$\text{Sol} \leftarrow Opt$$

For $i = 1$ to $k$
- Let $q_i \in S$ be the point that is farthest away from $H_{Sol \setminus \{o_i\}}$
- $\text{Sol} \leftarrow \text{Sol} \cup \{q_i\} \setminus \{o_i\}$
Let $V = \bigcup_i V_i$ be the union of the point sets

Let $S = \bigcup_i S_i$ be the union of core-sets

Let $Opt = \{o_1, ..., o_k\} \subseteq V$ be the optimal subset of points maximizing the volume

$$\text{Main Theorem}$$
Local Search produces a core-set for volume maximization

$$\text{Sol} \leftarrow Opt$$
For $i = 1$ to $k$
- Let $q_i \in S$ be the point that is farthest away from $H_{Sol \setminus \{o_i\}}$
- $\text{Sol} \leftarrow \text{Sol} \cup \{q_i\} \setminus \{o_i\}$
Main Theorem
Local Search produces a core-set for volume maximization

Let $V = \bigcup_i V_i$ be the union of the point sets
Let $S = \bigcup_i S_i$ be the union of core-sets
Let $Opt = \{o_1, \ldots, o_k\} \subset V$ be the optimal subset of points maximizing the volume

$Sol \leftarrow Opt$
For $i = 1$ to $k$
  • Let $q_i \in S$ be the point that is farthest away from $H_{S_i \setminus S_i}$
  • $Sol \leftarrow Sol \cup \{q_i\} \setminus \{o_i\}$

➤ Lose a factor of at most $2k$ at each iteration

Since local search preserve maximum distances to subspaces
Main Theorem
Local Search produces a core-set for volume maximization

Let $V = \bigcup_i V_i$ be the union of the point sets

Let $S = \bigcup_i S_i$ be the union of core-sets

Let $Opt = \{o_1, \ldots, o_k\} \subset V$ be the optimal subset of points maximizing the volume

$Sol \leftarrow Opt$

For $i = 1$ to $k$

- Let $q_i \in S$ be the point that is farthest away from $H_{Sol \setminus \{o_i\}}$
- $Sol \leftarrow Sol \cup \{q_i\} \setminus \{o_i\}$

➢ Lose a factor of at most $2k$ at each iteration

➢ Total approximation factor $(2k)^k$
Empirical Results

Data Set
• MNIST, with number of parts = 10
• MNIST, with number of parts = 50
• GENES, with number of parts = 10

Process
- Partition the data set randomly into parts
- Compute a core-set using one of the algorithms: Greedy, Local Search, LP-Based algorithm of [IMOR’18]
- Use greedy on the union of the coresets
Local Search vs Greedy

**Improvement of the solution of Local Search over Greedy**

- On average, 1.2%, 2.5%, and 9.6% improvement
- Some cases up to 58% improvement

**Ratio of runtime of Local Search over Greedy**

- On average, 6 times slower
Local Search vs. LP-based Algorithm of [IMOR’18]

Improvement of the solution of Local Search over [IMOR’18]
- On average, 1.4%, 1.8%, and 7.3% improvement
- Some cases up to 63% improvement

Ratio of runtime of Local Search over [IMOR’18]
- For lower values of k, Local Search is up to 50 times faster.
Summary

- Volume/Determinant Maximization Problem
- Notion of composable core-sets
- Algorithms that find composable core-sets for volume/determinant maximization

<table>
<thead>
<tr>
<th></th>
<th>[IMOR’18]</th>
<th>Greedy</th>
<th>Local Search</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approximation</td>
<td>$O(k \log k)^{k/2}$</td>
<td>$O(C^k^2)$</td>
<td>$O(k^k)$</td>
</tr>
<tr>
<td>Core-set Size</td>
<td>$O(k \log k)$</td>
<td>$k$</td>
<td>$k$</td>
</tr>
<tr>
<td>Simple?</td>
<td>×</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Empirical Approximation</td>
<td></td>
<td>Performs Best</td>
<td></td>
</tr>
<tr>
<td>Empirical Runtime</td>
<td>Slowest</td>
<td>Fastest</td>
<td>4 times slower than Greedy.</td>
</tr>
</tbody>
</table>
Summary
- Volume/Determinant Maximization Problem
- Notion of composable core-sets
- Algorithms that find composable core-sets for volume/determinant maximization

<table>
<thead>
<tr>
<th></th>
<th>[IMOR’18]</th>
<th>Greedy</th>
<th>Local Search</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approximation</td>
<td>$O(k \log k)^{\frac{k}{2}}$</td>
<td>$O(C^{k^2})$</td>
<td>$O(k^k)$</td>
</tr>
<tr>
<td>Core-set Size</td>
<td>$O(k \log k)$</td>
<td>$k$</td>
<td>$k$</td>
</tr>
<tr>
<td>Simple?</td>
<td>$\times$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
</tr>
<tr>
<td>Empirical Approximation</td>
<td>Performs Best</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Empirical Runtime</td>
<td>Slowest</td>
<td>Fastest</td>
<td>4 times slower than Greedy.</td>
</tr>
</tbody>
</table>

Conclusion
- Local Search might be the right algorithm to use in massive data models of computation.
Summary

- Volume/Determinant Maximization Problem
- Notion of composable core-sets
- Algorithms that find composable core-sets for volume/determinant maximization

<table>
<thead>
<tr>
<th></th>
<th>[IMOR’18]</th>
<th>Greedy</th>
<th>Local Search</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approximation</td>
<td>$O(k \log k)^{k/2}$</td>
<td>$O(C^{k^2})$</td>
<td>$O(k^k)$</td>
</tr>
<tr>
<td>Core-set Size</td>
<td>$O(k \log k)$</td>
<td>$k$</td>
<td>$k$</td>
</tr>
<tr>
<td>Simple?</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Empirical Approximation</td>
<td></td>
<td></td>
<td>Performs Best</td>
</tr>
<tr>
<td>Empirical Runtime</td>
<td>Slowest</td>
<td>Fastest</td>
<td>4 times slower than Greedy.</td>
</tr>
</tbody>
</table>

Conclusion

- Local Search might be the right algorithm to use in massive data models of computation.

Open Problem

- Tight analysis of Greedy: does it also provide approximation $k^{O(k)}$?
Summary

- Volume/Determinant Maximization Problem
- Notion of composable core-sets
- Algorithms that find composable core-sets for volume/determinant maximization

<table>
<thead>
<tr>
<th></th>
<th>[IMOR’18]</th>
<th>Greedy</th>
<th>Local Search</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approximation</td>
<td>$O(k \log k)^{k/2}$</td>
<td>$O(C^{k^2})$</td>
<td>$O(k^k)$</td>
</tr>
<tr>
<td>Core-set Size</td>
<td>$O(k \log k)$</td>
<td>$k$</td>
<td>$k$</td>
</tr>
<tr>
<td>Simple?</td>
<td>×</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Empirical Approximation</td>
<td></td>
<td></td>
<td>Performs Best</td>
</tr>
<tr>
<td>Empirical Runtime</td>
<td>Slowest</td>
<td>Fastest</td>
<td>4 times slower than Greedy.</td>
</tr>
</tbody>
</table>

Conclusion

- Local Search might be the right algorithm to use in massive data models of computation.

Open Problem

- Tight analysis of Greedy: does it also provide approximation $k^{O(k)}$?