A Fine-Grained Perspective on Approximating Subset Sum and Partition

Authors: Karl Bringmann and Vasileios Nakos
(Appeared in SODA'21)

Presented by Ce Jin

March 22, 2021

Overview

Subset Sum

Subset Sum

Given a (multi-)set X of n positive integers and a target integer t, compute $\mathrm{OPT} := \max\{\Sigma(Y) : Y \subseteq X, \Sigma(Y) \leq t\}$.

[1] $\tilde{O}(f)$ means $f \cdot (\log f)^{O(1)}$

Subset Sum

Subset Sum

Given a (multi-)set X of n positive integers and a target integer t, compute $\mathrm{OPT} := \max\{\Sigma(Y) : Y \subseteq X, \Sigma(Y) \leq t\}$.

Example: $X = \{1, 3, 4, 4\}$, $t = 6$

[1] $\tilde{O}(f)$ means $f \cdot (\log f)^{O(1)}$

Subset Sum

Subset Sum

Given a (multi-)set X of n positive integers and a target integer t, compute $\mathrm{OPT} := \max\{\Sigma(Y) : Y \subseteq X, \Sigma(Y) \leq t\}$.

Example: $X = \{1, 3, 4, 4\}$, $t = 6$
$\mathrm{OPT} = 1 + 4 = 5$

[1] $\tilde{O}(f)$ means $f \cdot (\log f)^{O(1)}$

Subset Sum

Subset Sum

Given a (multi-)set X of n positive integers and a target integer t, compute $\mathrm{OPT} := \max\{\Sigma(Y) : Y \subseteq X, \Sigma(Y) \leq t\}$.

Example: $X = \{1, 3, 4, 4\}$, $t = 6$
$\mathrm{OPT} = 1 + 4 = 5$

A classic NP-hard problem.

[1] $\tilde{O}(f)$ means $f \cdot (\log f)^{O(1)}$

Subset Sum

Subset Sum

Given a (multi-)set X of n positive integers and a target integer t, compute $\mathrm{OPT} := \max\{\Sigma(Y) : Y \subseteq X, \Sigma(Y) \leq t\}$.

Example: $X = \{1, 3, 4, 4\}$, $t = 6$
$\mathrm{OPT} = 1 + 4 = 5$

A classic NP-hard problem.

$O(2^{n/2})$ time algorithm [Horowitz and Sahni, JACM'74]

[1] $\tilde{O}(f)$ means $f \cdot (\log f)^{O(1)}$

Subset Sum

Subset Sum

Given a (multi-)set X of n positive integers and a target integer t, compute $\mathrm{OPT} := \max\{\Sigma(Y) : Y \subseteq X, \Sigma(Y) \leq t\}$.

Example: $X = \{1, 3, 4, 4\}$, $t = 6$
$\mathrm{OPT} = 1 + 4 = 5$

A classic NP-hard problem.

$O(2^{n/2})$ time algorithm [Horowitz and Sahni, JACM'74]
$\tilde{O}(n + t)$ time randomized algorithm[1] [Bringmann, SODA'17]

[1] $\tilde{O}(f)$ means $f \cdot (\log f)^{O(1)}$

Subset Sum

Subset Sum

Given a (multi-)set X of n positive integers and a target integer t, compute $\mathrm{OPT} := \max\{\Sigma(Y) : Y \subseteq X, \Sigma(Y) \leq t\}$.

Example: $X = \{1, 3, 4, 4\}$, $t = 6$
$\mathrm{OPT} = 1 + 4 = 5$

A classic NP-hard problem.

$O(2^{n/2})$ time algorithm [Horowitz and Sahni, JACM'74]
$\tilde{O}(n + t)$ time randomized algorithm[1] [Bringmann, SODA'17]

Partition Problem

A special case of Subset Sum where the input set X and target integer satisfy $t = \Sigma(X)/2$.

[1] $\tilde{O}(f)$ means $f \cdot (\log f)^{O(1)}$

Approximating Subset Sum

(Exact) Subset Sum

Given a (multi-)set X of n positive integers and a target integer t, compute $\mathrm{OPT} := \max\{\Sigma(Y) : Y \subseteq X, \Sigma(Y) \leq t\}$.

$(1-\varepsilon)$-approximate Subset Sum

Find a subset $Y \subseteq X$ such that

$$(1-\varepsilon) \cdot \mathrm{OPT} \leq \Sigma(Y) \leq t.$$

Approximating Subset Sum

(Exact) Subset Sum

Given a (multi-)set X of n positive integers and a target integer t, compute $\mathrm{OPT} := \max\{\Sigma(Y) : Y \subseteq X, \Sigma(Y) \leq t\}$.

$(1-\varepsilon)$-approximate Subset Sum

Find a subset $Y \subseteq X$ such that

$$(1-\varepsilon) \cdot \mathrm{OPT} \leq \Sigma(Y) \leq t.$$

There are algorithms in $\mathrm{poly}(n, 1/\varepsilon)$ time! ("FPTAS")

Approximating Subset Sum

(Exact) Subset Sum

Given a (multi-)set X of n positive integers and a target integer t, compute $\mathrm{OPT} := \max\{\Sigma(Y) : Y \subseteq X, \Sigma(Y) \leq t\}$.

$(1-\varepsilon)$-approximate Subset Sum

Find a subset $Y \subseteq X$ such that

$$(1-\varepsilon) \cdot \mathrm{OPT} \leq \Sigma(Y) \leq t.$$

There are algorithms in $\mathrm{poly}(n, 1/\varepsilon)$ time! ("FPTAS")
Previous best $\tilde{O}(\min\{n/\varepsilon, n + 1/\varepsilon^2\})$ time [Kellerer-Mansini-Pferschy-Speranza'03]

Approximating Subset Sum

(Exact) Subset Sum

Given a (multi-)set X of n positive integers and a target integer t, compute $\mathrm{OPT} := \max\{\Sigma(Y) : Y \subseteq X, \Sigma(Y) \leq t\}$.

$(1-\varepsilon)$-approximate Subset Sum

Find a subset $Y \subseteq X$ such that

$$(1-\varepsilon) \cdot \mathrm{OPT} \leq \Sigma(Y) \leq t.$$

There are algorithms in $\mathrm{poly}(n, 1/\varepsilon)$ time! ("FPTAS")
Previous best $\tilde{O}(\min\{n/\varepsilon, n + 1/\varepsilon^2\})$ time [Kellerer-Mansini-Pferschy-Speranza'03]

Approximating Subset Sum in $O((n + 1/\varepsilon)^{1.99})$ time?

Approximating Subset Sum

(Exact) Subset Sum

Given a (multi-)set X of n positive integers and a target integer t, compute $\mathrm{OPT} := \max\{\Sigma(Y) : Y \subseteq X, \Sigma(Y) \leq t\}$.

$(1-\varepsilon)$-approximate Subset Sum

Find a subset $Y \subseteq X$ such that

$$(1-\varepsilon) \cdot \mathrm{OPT} \leq \Sigma(Y) \leq t.$$

There are algorithms in $\mathrm{poly}(n, 1/\varepsilon)$ time! ("FPTAS")
Previous best $\tilde{O}(\min\{n/\varepsilon, n + 1/\varepsilon^2\})$ time [Kellerer-Mansini-Pferschy-Speranza'03]

Approximating Subset Sum in $O((n+1/\varepsilon)^{1.99})$ time?

Partition: $\tilde{O}(n + 1/\varepsilon^{5/3})$ (randomized) [Mucha-Węgrzycki-Włodarczyk SODA'19]

Main results in this paper (1)

Conditional hardness of approximating Subset Sum

If $(\min, +)$-convolution requires $n^{2-o(1)}$ time, then $(1-\varepsilon)$-approximating Subset Sum requires $(n + 1/\varepsilon)^{2-o(1)}$ time.

$(\min, +)$-convolution: Given $a[1..n], b[1..n]$, compute $c[i] = \min_{j+k=i} a[j] + b[k]$ for all i.

Main results in this paper (1)

Conditional hardness of approximating Subset Sum

If $(\min,+)$-convolution requires $n^{2-o(1)}$ time, then $(1-\varepsilon)$-approximating Subset Sum requires $(n+1/\varepsilon)^{2-o(1)}$ time.

$(\min,+)$-convolution: Given $a[1..n], b[1..n]$, compute $c[i] = \min_{j+k=i} a[j] + b[k]$ for all i.

Previously, such a lower bound was only known for the harder problem of approximating Knapsack. [Cygan-Mucha-Węgrzycki-Włodarczyk'17, Künnemann-Paturi-Schneider'17]

Main results in this paper (2)

Reduction from Subset Sum to $(\min, +)$-convolution

If $(\min, +)$-convolution can be solved in $T(n)$ time, then Subset Sum can be $(1-\varepsilon)$-approximated w.h.p.[a] in $\tilde{O}(n + T(1/\varepsilon))$ time.

[a] with high probability, i.e., 1-$1/n^C$ for arbitrary constant C.

($T(n) \leq n^2/2^{\Omega(\sqrt{\log n})}$ [Williams'14, Chan-Williams'16])

Main results in this paper (2)

Reduction from Subset Sum to $(\min, +)$-convolution

If $(\min, +)$-convolution can be solved in $T(n)$ time, then Subset Sum can be $(1-\varepsilon)$-approximated w.h.p.[a] in $\tilde{O}(n + T(1/\varepsilon))$ time.

[a] with high probability, i.e., 1-$1/n^C$ for arbitrary constant C.

($T(n) \le n^2/2^{\Omega(\sqrt{\log n})}$ [Williams'14, Chan-Williams'16])

Improves the previous algorithm by a super-polylog-factor.

Main results in this paper (2)

Reduction from Subset Sum to $(\min, +)$-convolution

If $(\min, +)$-convolution can be solved in $T(n)$ time, then Subset Sum can be $(1-\varepsilon)$-approximated w.h.p.[a] in $\tilde{O}(n + T(1/\varepsilon))$ time.

[a] with high probability, i.e., 1-$1/n^C$ for arbitrary constant C.

($T(n) \leq n^2/2^{\Omega(\sqrt{\log n})}$ [Williams'14, Chan-Williams'16])

Improves the previous algorithm by a super-polylog-factor.

$(\min, +)$-convolution and approximating Subset Sum are *fine-grained equivalent*!

Main results in this paper (3)

Better approximation algorithm for Partition

Partition can be approximated deterministically in $\tilde{O}(n + (1/\varepsilon)^{3/2}/2^{\Omega(\sqrt{\log 1/\varepsilon})})$ time.

Improves the previous $\tilde{O}(n + (1/\varepsilon)^{5/3})$-time randomized algorithm.

Conditional lower bound

Knapsack

(Exact) Knapsack

Given a set X of n items each having weight $w_i \in \mathbb{N}^+$ and value $v_i \in \mathbb{N}^+$, and a capacity $W \in \mathbb{N}^+$, compute $\max\{v(Y) : Y \subseteq X, w(Y) \leq W\}$, where $v(Y)$ (and $w(Y)$) denote the total value (and weight) of items in Y.

We know $O(nW)$ or $\tilde{O}(n + W^2)$ algorithms (quadratic time).

Knapsack

(Exact) Knapsack

Given a set X of n items each having weight $w_i \in \mathbb{N}^+$ and value $v_i \in \mathbb{N}^+$, and a capacity $W \in \mathbb{N}^+$, compute $\max\{v(Y) : Y \subseteq X, w(Y) \leq W\}$, where $v(Y)$ (and $w(Y)$) denote the total value (and weight) of items in Y.

We know $O(nW)$ or $\tilde{O}(n + W^2)$ algorithms (quadratic time).

From $(\min, +)$-convolution to Knapsack [Cygan et al.'17, Künnemann et al.'17]

A $T(n, W)$-time algorithm for Knapsack would imply a $\tilde{O}(T(\sqrt{n}, \sqrt{n}) \cdot n)$-time algorithm for $(\min, +)$-convolution.

Knapsack

(Exact) Knapsack

Given a set X of n items each having weight $w_i \in \mathbb{N}^+$ and value $v_i \in \mathbb{N}^+$, and a capacity $W \in \mathbb{N}^+$, compute $\max\{v(Y) : Y \subseteq X, w(Y) \leq W\}$, where $v(Y)$ (and $w(Y)$) denote the total value (and weight) of items in Y.

We know $O(nW)$ or $\tilde{O}(n + W^2)$ algorithms (quadratic time).

From $(\min, +)$-convolution to Knapsack [Cygan et al.'17, Künnemann et al.'17]

A $T(n, W)$-time algorithm for Knapsack would imply a $\tilde{O}(T(\sqrt{n}, \sqrt{n}) \cdot n)$-time algorithm for $(\min, +)$-convolution.

From Knapsack to Apx-SubsetSum (this paper)

A $T(n, 1/\varepsilon)$-time algorithm for approximating Subset Sum would imply a $\tilde{O}(T(n, W) + W)$-time algorithm for Knapsack.

Reduction from Knapsack

Decision version of Knapsack

Given a set X of n items each having weight w_i and value v_i, and a capacity W and a target total value V.
Is there a $Y \subseteq X$ such that $w(Y) \leq W$ and $v(Y) \geq V$?

Reduction from Knapsack

Decision version of Knapsack

Given a set X of n items each having weight w_i and value v_i, and a capacity W and a target total value V.
Is there a $Y \subseteq X$ such that $w(Y) \leq W$ and $v(Y) \geq V$?

Assume all input integers are in $\{1, 2, \ldots, M\}$. Let $M' = 4nM$.

1. Add items with $w_i = 0$ and $v_i = -1, -2, -4, , \ldots, -2^{\log M}$.
   Add items with $v_i = 0$ and $w_i = 1, 2, 4, \ldots, 2^{\log M}$.

Reduction from Knapsack

Decision version of Knapsack

Given a set X of n items each having weight w_i and value v_i, and a capacity W and a target total value V.
Is there a $Y \subseteq X$ such that $w(Y) \leq W$ and $v(Y) \geq V$?

Assume all input integers are in $\{1, 2, \ldots, M\}$. Let $M' = 4nM$.

1. Add items with $w_i = 0$ and $v_i = -1, -2, -4, , \ldots, -2^{\log M}$.
   Add items with $v_i = 0$ and $w_i = 1, 2, 4, \ldots, 2^{\log M}$. Now we can replace "$w(Y) \leq W$ and $v(Y) \geq V$" by "$w(Y) = W$ and $v(Y) = V$"!

Reduction from Knapsack

Decision version of Knapsack

Given a set X of n items each having weight w_i and value v_i, and a capacity W and a target total value V.
Is there a $Y \subseteq X$ such that $w(Y) \leq W$ and $v(Y) \geq V$?

Assume all input integers are in $\{1, 2, \ldots, M\}$. Let $M' = 4nM$.

1. Add items with $w_i = 0$ and $v_i = -1, -2, -4, , \ldots, -2^{\log M}$.
   Add items with $v_i = 0$ and $w_i = 1, 2, 4, \ldots, 2^{\log M}$. Now we can replace "$w(Y) \leq W$ and $v(Y) \geq V$" by "$w(Y) = W$ and $v(Y) = V$"!
2. Define the Apx-Subset Sum instance: $t := WM' - V$, $\varepsilon = 1/(2W)$, and $x_i := w_i \cdot M' - v_i$.

Reduction from Knapsack

Decision version of Knapsack

Given a set X of n items each having weight w_i and value v_i, and a capacity W and a target total value V.
Is there a $Y \subseteq X$ such that $w(Y) \leq W$ and $v(Y) \geq V$?

Assume all input integers are in $\{1, 2, \ldots, M\}$. Let $M' = 4nM$.

1. Add items with $w_i = 0$ and $v_i = -1, -2, -4, , \ldots, -2^{\log M}$.
   Add items with $v_i = 0$ and $w_i = 1, 2, 4, \ldots, 2^{\log M}$. Now we can replace "$w(Y) \leq W$ and $v(Y) \geq V$" by "$w(Y) = W$ and $v(Y) = V$"!
2. Define the Apx-Subset Sum instance: $t := WM' - V$, $\varepsilon = 1/(2W)$, and $x_i := w_i \cdot M' - v_i$.

Knapsack has a solution $\Longrightarrow$ Subset Sum has an exact solution.

Reduction from Knapsack

Decision version of Knapsack

Given a set X of n items each having weight w_i and value v_i, and a capacity W and a target total value V.
Is there a $Y \subseteq X$ such that $w(Y) \leq W$ and $v(Y) \geq V$?

Assume all input integers are in $\{1, 2, \ldots, M\}$. Let $M' = 4nM$.

1. Add items with $w_i = 0$ and $v_i = -1, -2, -4, , \ldots, -2^{\log M}$.
   Add items with $v_i = 0$ and $w_i = 1, 2, 4, \ldots, 2^{\log M}$. Now we can replace "$w(Y) \leq W$ and $v(Y) \geq V$" by "$w(Y) = W$ and $v(Y) = V$"!
2. Define the Apx-Subset Sum instance: $t := WM' - V$, $\varepsilon = 1/(2W)$, and $x_i := w_i \cdot M' - v_i$.

Knapsack has a solution $\implies$ Subset Sum has an exact solution.
Knapsack has no solution $\implies$ Every subset sum is either $> t$ or $< (1 - \varepsilon)t$.

Algorithm for Partition

Partition

$(1-\varepsilon)$-Approximating Partition

Given a (multi-)set X of n positive integers, let
$\mathrm{OPT} := \max\{\Sigma(Y) : Y \subseteq X, \Sigma(Y) \leq t\}$, where $t = \Sigma(X)/2$.
Find a subset $Y \subseteq X$ such that $(1-\varepsilon) \cdot \mathrm{OPT} \leq \Sigma(Y) \leq t$.

Sumset

For $A, B \subseteq \mathbb{N}$, define

$$A + B = \{a + b : a \in A, b \in B\}.$$

Sumset

For $A, B \subseteq \mathbb{N}$, define

$$A + B = \{a + b : a \in A, b \in B\}.$$

Exact algorithm: compute $\{0, x_1\} + \{0, x_2\} + \cdots + \{0, x_n\}$. (the sumset is too big!)

Sumset

For $A, B \subseteq \mathbb{N}$, define

$$A + B = \{a + b : a \in A, b \in B\}.$$

Exact algorithm: compute $\{0, x_1\} + \{0, x_2\} + \cdots + \{0, x_n\}$. (the sumset is too big!)

Approximation algorithm: compute an approximate version of the sumset (which can be sparse)

A notion of approximation

Approximation

For $b \in \mathbb{N}$ and $A \subseteq \mathbb{N}$, define

$$apx^-(b, A) := \max\{a \in A : a \leq b\}$$

$$apx^+(b, A) := \min\{a \in A : a \geq b\}$$

We say A Δ-approximates B if $A \subseteq B$ and for every $b \in B$,

$$apx^+(b, A) - apx^-(b, A) \leq \Delta.$$

[2] Inspired by Kellerer et al.'03

A notion of approximation

Approximation

For $b \in \mathbb{N}$ and $A \subseteq \mathbb{N}$, define

$$apx^-(b, A) := \max\{a \in A : a \leq b\}$$
$$apx^+(b, A) := \min\{a \in A : a \geq b\}$$

We say A Δ-approximates B if $A \subseteq B$ and for every $b \in B$,

$$apx^+(b, A) - apx^-(b, A) \leq \Delta.$$

This[2] is stronger than the usual definition of $apx^+(b, A) - b \leq \Delta$ and $b - apx^-(b, A) \leq \Delta$.

[2] Inspired by Kellerer et al.'03

A notion of approximation

Approximation

For $b \in \mathbb{N}$ and $A \subseteq \mathbb{N}$, define

$$apx^-(b, A) := \max\{a \in A : a \leq b\}$$

$$apx^+(b, A) := \min\{a \in A : a \geq b\}$$

We say A Δ-approximates B if $A \subseteq B$ and for every $b \in B$,

$$apx^+(b, A) - apx^-(b, A) \leq \Delta.$$

This[2] is stronger than the usual definition of $apx^+(b, A) - b \leq \Delta$ and $b - apx^-(b, A) \leq \Delta$.

Sumset property

If A_1 Δ-approximates B_1 and A_2 Δ-approximates B_2, then $A_1 + A_2$ Δ-approximates $B_1 + B_2$.

Notice that the approximation error Δ doesn't blow up to 2Δ!

[2] Inspired by Kellerer et al.'03

A notion of approximation

Transitivity

If A Δ-approximates B, and B Δ-approximates C, then A Δ-approximates C.

A notion of approximation

Transitivity

If A Δ-approximates B, and B Δ-approximates C, then A Δ-approximates C.

Sparsification

Given a set $B \subseteq [t]$ [a], we can compute in linear time a set A that *sparsely* Δ-approximates B, where $|A| \leq O((t/\Delta) + 1)$.

[a] Here we define $[t] = \{0, 1, 2, \ldots, t\}$

A notion of approximation

Transitivity

If A Δ-approximates B, and B Δ-approximates C, then A Δ-approximates C.

Sparsification

Given a set $B \subseteq [t]$ [a], we can compute in linear time a set A that *sparsely* Δ-approximates B, where $|A| \leq O((t/\Delta) + 1)$.

[a] Here we define $[t] = \{0, 1, 2, \ldots, t\}$

Approximate Sumset computation:
If A_1 sparsely Δ-approximates $B_1 \subseteq [t]$, A_2 sparsely Δ-approximates $B_2 \subseteq [t]$, then we can compute a sparse Δ-approximation of $B_1 + B_2$ in $O(T_{minconv}(t/\Delta))$ time. $(T_{minconv}(n) \leq n^2/2^{\Omega(\sqrt{\log n})})$

A simple algorithm for Partition

Recall $t = \Sigma(X)/2$.
Every non-trivial instance satisfies $\mathrm{OPT} \geq \Sigma(X)/4$.

A simple algorithm for Partition

Recall $t = \Sigma(X)/2$.
Every non-trivial instance satisfies $\mathrm{OPT} \geq \Sigma(X)/4$.
Set $\Delta = \varepsilon\Sigma(X)/4$.
Goal: compute a sparse Δ-approximation of $\{0, x_1\} + \cdots + \{0, x_n\}$.

A simple algorithm for Partition

Recall $t = \Sigma(X)/2$.
Every non-trivial instance satisfies $\mathrm{OPT} \geq \Sigma(X)/4$.
Set $\Delta = \varepsilon\Sigma(X)/4$.
Goal: compute a sparse Δ-approximation of $\{0, x_1\} + \cdots + \{0, x_n\}$.

Algorithm: Use approximate sumset computation to merge in a binary-tree-like fashion.
The total time in each level is at most $O(n + (\Sigma(X)/\Delta)^2) = O(n + 1/\varepsilon^2)$

Improve using FFT

Recall $t = \Sigma(X)/2$.
Set $\Delta = \varepsilon\Sigma(X)/4$.

Improve using FFT

Recall $t = \Sigma(X)/2$.
Set $\Delta = \varepsilon\Sigma(X)/4$.
Choose a parameter $1 \leq L \leq 1/\varepsilon$

Divide $\{x_1, x_2, \ldots, x_n\}$ into L groups of balanced sizes.

Improve using FFT

Recall $t = \Sigma(X)/2$.
Set $\Delta = \varepsilon\Sigma(X)/4$.
Choose a parameter $1 \leq L \leq 1/\varepsilon$

Divide $\{x_1, x_2, \ldots, x_n\}$ into L groups of balanced sizes.
Compute $Z_1, \ldots, Z_L$ which Δ-approximate the sumsets of these groups

Improve using FFT

Recall $t = \Sigma(X)/2$.
Set $\Delta = \varepsilon\Sigma(X)/4$.
Choose a parameter $1 \le L \le 1/\varepsilon$

Divide $\{x_1, x_2, \dots, x_n\}$ into L groups of balanced sizes.
Compute $Z_1, \dots, Z_L$ which Δ-approximate the sumsets of these groups

Round every integer in Z_i down to the nearest integer multiple of Δ/L.

Improve using FFT

Recall $t = \Sigma(X)/2$.
Set $\Delta = \varepsilon\Sigma(X)/4$.
Choose a parameter $1 \leq L \leq 1/\varepsilon$

Divide $\{x_1, x_2, \ldots, x_n\}$ into L groups of balanced sizes.
Compute $Z_1, \ldots, Z_L$ which Δ-approximate the sumsets of these groups

Round every integer in Z_i down to the nearest integer multiple of Δ/L.
Use FFT to merge these sumsets in a binary-tree-like fashion

Improve using FFT

Recall $t = \Sigma(X)/2$.
Set $\Delta = \varepsilon\Sigma(X)/4$.
Choose a parameter $1 \leq L \leq 1/\varepsilon$

Divide $\{x_1, x_2, \ldots, x_n\}$ into L groups of balanced sizes.
Compute $Z_1, \ldots, Z_L$ which Δ-approximate the sumsets of these groups

Round every integer in Z_i down to the nearest integer multiple of Δ/L.
Use FFT to merge these sumsets in a binary-tree-like fashion

Because of rounding, the values in the final sumset may not be realizable.
Each y in the final sumset corresponds to a subset $Y \subseteq X$ with
$\Sigma(Y) \in [y, y + L \cdot (\Delta/L)] = [y, y + \Delta]$.

Improve using FFT

Recall $t = \Sigma(X)/2$.
Set $\Delta = \varepsilon\Sigma(X)/4$.
Choose a parameter $1 \leq L \leq 1/\varepsilon$

Divide $\{x_1, x_2, \ldots, x_n\}$ into L groups of balanced sizes.
Compute $Z_1, \ldots, Z_L$ which Δ-approximate the sumsets of these groups

Round every integer in Z_i down to the nearest integer multiple of Δ/L.
Use FFT to merge these sumsets in a binary-tree-like fashion

Because of rounding, the values in the final sumset may not be realizable.
Each y in the final sumset corresponds to a subset $Y \subseteq X$ with $\Sigma(Y) \in [y, y + L \cdot (\Delta/L)] = [y, y + \Delta]$.

Issue: what if $y \leq t$ but $\Sigma(Y) > t$ (an invalid subset)?

Improve using FFT

Recall $t = \Sigma(X)/2$.
Set $\Delta = \varepsilon\Sigma(X)/4$.
Choose a parameter $1 \leq L \leq 1/\varepsilon$

Divide $\{x_1, x_2, \ldots, x_n\}$ into L groups of balanced sizes.
Compute $Z_1, \ldots, Z_L$ which Δ-approximate the sumsets of these groups

Round every integer in Z_i down to the nearest integer multiple of Δ/L.
Use FFT to merge these sumsets in a binary-tree-like fashion

Because of rounding, the values in the final sumset may not be realizable.
Each y in the final sumset corresponds to a subset $Y \subseteq X$ with $\Sigma(Y) \in [y, y + L \cdot (\Delta/L)] = [y, y + \Delta]$.

Issue: what if $y \leq t$ but $\Sigma(Y) > t$ (an invalid subset)?
Take $Y' = X - Y$ with $\Sigma(Y') = \Sigma(X) - Y \geq \Sigma(X) - (t + \Delta) = t - \Delta$.

Time complexity

Divide $\{x_1, x_2, \dots, x_n\}$ into L groups of balanced sizes $s_1 + s_2 + \cdots + s_L = \Sigma(X), s_i \approx \Sigma(X)/L$.

Time complexity

Divide $\{x_1, x_2, \ldots, x_n\}$ into L groups of balanced sizes $s_1 + s_2 + \cdots + s_L = \Sigma(X), s_i \approx \Sigma(X)/L$.
Compute $Z_1, \ldots, Z_L$ which Δ-approximate the sumsets of these groups, in $\approx L \cdot (s_i/\Delta)^2 \approx L \cdot (\frac{1}{\varepsilon L})^2$ total time.

Time complexity

Divide $\{x_1, x_2, \ldots, x_n\}$ into L groups of balanced sizes $s_1 + s_2 + \cdots + s_L = \Sigma(X), s_i \approx \Sigma(X)/L$.
Compute $Z_1, \ldots, Z_L$ which Δ-approximate the sumsets of these groups, in $\approx L \cdot (s_i/\Delta)^2 \approx L \cdot (\frac{1}{\varepsilon L})^2$ total time.

Round every integer in Z_i down to the nearest integer multiple of Δ/L.
Use FFT to merge these sumsets in a binary-tree-like fashion in $\tilde{O}(\sum_{1\le i\le L} \frac{s_i}{\Delta/L}) = \tilde{O}(L/\varepsilon)$ total time.

Time complexity

Divide $\{x_1, x_2, \ldots, x_n\}$ into L groups of balanced sizes
$s_1 + s_2 + \cdots + s_L = \Sigma(X)$, $s_i \approx \Sigma(X)/L$.
Compute $Z_1, \ldots, Z_L$ which Δ-approximate the sumsets of these groups, in $\approx L \cdot (s_i/\Delta)^2 \approx L \cdot (\frac{1}{\varepsilon L})^2$ total time.

Round every integer in Z_i down to the nearest integer multiple of Δ/L.
Use FFT to merge these sumsets in a binary-tree-like fashion in
$\tilde{O}(\sum_{1 \le i \le L} \frac{s_i}{\Delta/L}) = \tilde{O}(L/\varepsilon)$ total time.

Choose $L \approx 1/\varepsilon^{1/2}$, the total time is $\approx n + 1/\varepsilon^{3/2}$.

Open problems

This paper settled the complexity of approximating **Subset Sum**. There are a few other open problems.

Open problems

This paper settled the complexity of approximating **Subset Sum**. There are a few other open problems.

Faster approximation algorithms for **Partition**? Or better lower bounds?
No poly(n)$/\varepsilon^{1-\Omega(1)}$-time algorithms exist, assuming SETH.
[Abboud-Bringmann-Hermelin-Shabtay'19]
Upper bound: $\tilde{O}(n + \varepsilon^{1.5})$ [Bringmann-Nakos'21 (this paper)]

Open problems

This paper settled the complexity of approximating **Subset Sum**. There are a few other open problems.

Faster approximation algorithms for **Partition**? Or better lower bounds?
No poly(n)/$\varepsilon^{1-\Omega(1)}$-time algorithms exist, assuming SETH.
[Abboud-Bringmann-Hermelin-Shabtay'19]
Upper bound: $\tilde{O}(n + \varepsilon^{1.5})$ [Bringmann-Nakos'21 (this paper)]

Faster approximation algorithms for **Knapsack**?
No $(n + 1/\varepsilon)^{2-\Omega(1)}$-time algorithms exist, assuming $(\min, +)$-convolution conjecture. [Cygan et al.'17, Künnemann et al.'17]
Upper bound: $\tilde{O}(n + \varepsilon^{2.25})$ [Jin'19]

Open problems

This paper settled the complexity of approximating **Subset Sum**. There are a few other open problems.

Faster approximation algorithms for **Partition**? Or better lower bounds?
No poly(n)/$\varepsilon^{1-\Omega(1)}$-time algorithms exist, assuming SETH.
[Abboud-Bringmann-Hermelin-Shabtay'19]
Upper bound: $\tilde{O}(n + \varepsilon^{1.5})$ [Bringmann-Nakos'21 (this paper)]

Faster approximation algorithms for **Knapsack**?
No $(n + 1/\varepsilon)^{2-\Omega(1)}$-time algorithms exist, assuming (min, +)-convolution conjecture. [Cygan et al.'17, Künnemann et al.'17]
Upper bound: $\tilde{O}(n + \varepsilon^{2.25})$ [Jin'19]

Thanks!