Pattern Matching and Consensus Problems on Weighted Sequences and Profiles

Tomasz Kociumaka[1], Solon P. Pissis[2], Jakub Radoszewski[1,2]

[1]University of Warsaw, Poland

[2]King's College London, UK

ISAAC 2016

Sydney, Australia

December 14, 2016

Strings, Partial Words, and Indeterminate Strings

Strings (solid strings):

a c a b b b

Strings, Partial Words, and Indeterminate Strings

Strings (solid strings):

a c a b b b

Partial words (strings with don't care symbols):

a $\diamondsuit$ a b $\diamondsuit$ b

Strings, Partial Words, and Indeterminate Strings

Strings (solid strings):

a	c	a	b	b	b

Partial words (strings with don't care symbols):

a	$\diamondsuit$	a	b	$\diamondsuit$	b
a	c	a	b	b	b
a	a	a	b	a	b
a	b	a	b	c	b

$\vdots$

Strings, Partial Words, and Indeterminate Strings

Strings (solid strings):

a	c	a	b	b	b

Partial words (strings with don't care symbols):

a	$\diamondsuit$	a	b	$\diamondsuit$	b
a	c	a	b	b	b
a	a	a	b	a	b
a	b	a	b	c	b

$\vdots$

Indeterminate strings:

a	b c	a	b	a b	b

Strings (solid strings):

a	c	a	b	b	b

Partial words (strings with don't care symbols):

a	$\diamondsuit$	a	b	$\diamondsuit$	b
a	c	a	b	b	b
a	a	a	b	a	b
a	b	a	b	c	b

$\vdots$

Indeterminate strings:

a	b c	a	b	a b	b
a	c	a	b	b	b
a	c	a	b	a	b
a	b	a	b	b	b
a	b	a	b	a	b

Weighted Sequences (PPMs) and Profiles (PWMs)

Weighted Sequences (Position Probability Matrices):

$$\mathtt{a}\quad \begin{matrix}\mathtt{b}\ 0.2\\ \mathtt{c}\ 0.8\end{matrix}\quad \mathtt{a}\quad \mathtt{b}\quad \begin{matrix}\mathtt{a}\ 0.6\\ \mathtt{b}\ 0.4\end{matrix}\quad \mathtt{b}$$

Weighted Sequences (PPMs) and Profiles (PWMs)

Weighted Sequences (Position Probability Matrices):

a	b 0.2 c 0.8	a	b	a 0.6 b 0.4	b	probability
a	c	a	b	b	b	0.32
a	c	a	b	a	b	0.48
a	b	a	b	b	b	0.08
a	b	a	b	a	b	0.12

Weighted Sequences (PPMs) and Profiles (PWMs)

Weighted Sequences (Position Probability Matrices):

a	b 0.2 c 0.8	a	b	a 0.6 b 0.4	b	probability
a	c	a	b	b	b	0.32
a	c	a	b	a	b	0.48
a	b	a	b	b	b	0.08
a	b	a	b	a	b	0.12

Weighted Sequences (PPMs) and Profiles (PWMs)

Weighted Sequences (Position Probability Matrices):

a 1	a 0	a 1	a 0	a 0.6	a 0	
b 0	b 0.2	b 0	b 1	b 0.4	b 1	
c 0	c 0.8	c 0	c 0	c 0	c 0	probability
a	c	a	b	b	b	0.32
a	c	a	b	a	b	0.48
a	b	a	b	b	b	0.08
a	b	a	b	a	b	0.12

Weighted Sequences (PPMs) and Profiles (PWMs)

Weighted Sequences (Position Probability Matrices):

a 1 b 0 c 0	a 0 b 0.2 c 0.8	a 1 b 0 c 0	a 0 b 1 c 0	a 0.6 b 0.4 c 0	a 0 b 1 c 0	probability
a	c	a	b	b	b	0.32
a	c	a	b	a	b	0.48
a	b	a	b	b	b	0.08
a	b	a	b	a	b	0.12

Profiles (Position Weight Matrices):

a 7 b 0 c 1	a 3 b 2 c 8	a 0 b 1 c 0	a 0 b 5 c 0	a 6 b 4 c 3	a 1 b 9 c 0

Weighted Sequences (PPMs) and Profiles (PWMs)

Weighted Sequences (Position Probability Matrices):

a 1	a 0	a 1	a 0	a 0.6	a 0	
b 0	b 0.2	b 0	b 1	b 0.4	b 1	probability
c 0	c 0.8	c 0	c 0	c 0	c 0	
a	c	a	b	b	b	0.32
a	c	a	b	a	b	0.48
a	b	a	b	b	b	0.08
a	b	a	b	a	b	0.12

Profiles (Position Weight Matrices):

a 7	a 3	a 0	a 0	a 6	a 1	
b 0	b 2	b 1	b 5	b 4	b 9	score
c 1	c 8	c 0	c 0	c 3	c 0	
a	c	a	b	b	b	33
a	c	a	b	a	b	35
a	b	a	b	b	b	27
a	b	a	b	a	b	29

⋮

Weighted Sequences (PPMs) and Profiles (PWMs)

Weighted Sequences (Position Probability Matrices): ← **this talk**

a 1 b 0 c 0	a 0 b 0.2 c 0.8	a 1 b 0 c 0	a 0 b 1 c 0	a 0.6 b 0.4 c 0	a 0 b 1 c 0	probability
a	c	a	b	b	b	0.32
a	c	a	b	a	b	0.48
a	b	a	b	b	b	0.08
a	b	a	b	a	b	0.12

Profiles (Position Weight Matrices): ← **ask me or see our paper**

a 7 b 0 c 1	a 3 b 2 c 8	a 0 b 1 c 0	a 0 b 5 c 0	a 6 b 4 c 3	a 1 b 9 c 0	score
a	c	a	b	b	b	33
a	c	a	b	a	b	35
a	b	a	b	b	b	27
a	b	a	b	a	b	29

⋮

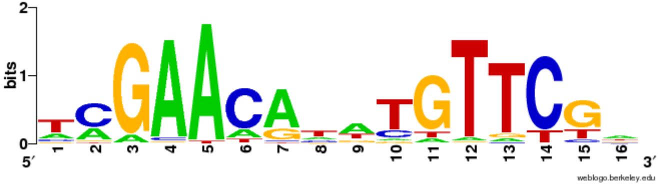
Applications of Uncertain Strings

Bioinformatics

- introduced in:

  Stormo, Schneider, Gold, and Ehrenfeucht (1982). "Use of the 'Perceptron' algorithm to distinguish translational initiation sites in E. coli". Nucleic Acids Research **10** (9): 2997–3011.

- one of the standard representations of motifs



Source: Gnomehacker at English Wikipedia [GFDL, CC BY-SA 3.0] via Wikimedia Commons

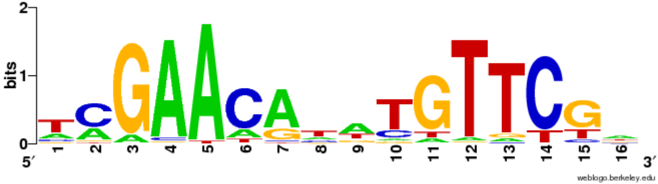
Applications of Uncertain Strings

Bioinformatics

- introduced in:

  Stormo, Schneider, Gold, and Ehrenfeucht (1982). "Use of the 'Perceptron' algorithm to distinguish translational initiation sites in E. coli". Nucleic Acids Research **10** (9): 2997–3011.

- one of the standard representations of motifs



Source: Gnomehacker at English Wikipedia [GFDL, CC BY-SA 3.0] via Wikimedia Commons

Other applications

- annotation of noisy sensor data,
- probabilistic databases.

Threshold

- Score function and probability distribution are defined on all solid strings of matching length.
- Typically, only high values are considered significant.

Threshold

- Score function and probability distribution are defined on all solid strings of matching length.
- Typically, only high values are considered significant.

Definition

A solid string S matches a weighed sequence X if $\mathcal{P}(S, X) \geq \frac{1}{z}$ for a given threshold $\frac{1}{z}$.

Threshold

- Score function and probability distribution are defined on all solid strings of matching length.
- Typically, only high values are considered significant.

Definition

A solid string S matches a weighed sequence X if $\mathcal{P}(S, X) \geq \frac{1}{z}$ for a given threshold $\frac{1}{z}$.

Fact

For every weighted sequence X, there are at most z strings X satisfying $\mathcal{P}(S, X) \geq \frac{1}{z}$.

Threshold

- Score function and probability distribution are defined on all solid strings of matching length.
- Typically, only high values are considered significant.

Definition

A solid string S matches a weighed sequence X if $\mathcal{P}(S,X) \geq \frac{1}{z}$ for a given threshold $\frac{1}{z}$.

Fact

For every weighted sequence X, there are at most z strings X satisfying $\mathcal{P}(S,X) \geq \frac{1}{z}$.

- z is used as a parameter for designing the algorithms.

Algorithms for Weighted Sequences

Indexing and pattern matching

Christodoulakis et al., 2004; Iliopoulos et al., 2006;
Amir et al., 2008; Barton et al., 2016; Biswas et al., 2016

Approximate and gapped pattern matching

Zhang et al., 2004; Amir et al., 2006; Zhang et al., 2010

Repetitions and regularities discovery

Iliopoulos et al., 2005; Christodoulakis et al., 2006;
Zhang et al., 2013; Barton and Pissis, 2014

Longest common subsequence problem

Amir et al., 2009; Cygan et al., 2011

Alignment of weighted sequences

Na et al., 2009

Algorithms for Weighted Sequences

Indexing and **pattern matching** $\leftarrow$ **this work**

Christodoulakis et al., 2004; Iliopoulos et al., 2006;

Amir et al., 2008; Barton et al., 2016; Biswas et al., 2016

Approximate and gapped pattern matching

Zhang et al., 2004; Amir et al., 2006; Zhang et al., 2010

Repetitions and regularities discovery

Iliopoulos et al., 2005; Christodoulakis et al., 2006;

Zhang et al., 2013; Barton and Pissis, 2014

Longest common subsequence problem

Amir et al., 2009; Cygan et al., 2011

Alignment of weighted sequences

Na et al., 2009

Consensus and generalized pattern matching $\leftarrow$ **this work**

Weighted Pattern Matching

Weighted Pattern Matching

Given a weighted sequence T, a threshold $1/z$, and a solid pattern P, determine all the fragments of T matching P.

Weighted Pattern Matching

Weighted Pattern Matching

Given a weighted sequence T, a threshold $1/z$, and a solid pattern P, determine all the fragments of T matching P.

$\frac{1}{z} = 0.2$ T :

a 0.25	a 1	a 0.75	a 0.5	a 1
b 0.75	b 0	b 0.25	b 0.5	b 0

Weighted Pattern Matching

Weighted Pattern Matching

Given a weighted sequence T, a threshold $1/z$, and a solid pattern P, determine all the fragments of T matching P.

$\frac{1}{z} = 0.2$

T :	a 0.25 b 0.75	a 1 b 0	a 0.75 b 0.25	a 0.5 b 0.5	a 1 b 0
P :	a	a	b		

Weighted Pattern Matching

Given a weighted sequence T, a threshold $1/z$, and a solid pattern P, determine all the fragments of T matching P.

$\frac{1}{z} = 0.2$	T :	a 0.25 b 0.75	a 1 b 0	a 0.75 b 0.25	a 0.5 b 0.5	a 1 b 0
$0.0625 < \frac{1}{z}$	P :	a	a	b		

Weighted Pattern Matching

Weighted Pattern Matching

Given a weighted sequence T, a threshold $1/z$, and a solid pattern P, determine all the fragments of T matching P.

$\frac{1}{z} = 0.2$

T :	a 0.25 b 0.75	a 1 b 0	a 0.75 b 0.25	a 0.5 b 0.5	a 1 b 0
P :		a	a	b	

Weighted Pattern Matching

WEIGHTED PATTERN MATCHING

Given a weighted sequence T, a threshold $1/z$, and a solid pattern P, determine all the fragments of T matching P.

$\frac{1}{z} = 0.2$	T :	a 0.25 b 0.75	a 1 b 0	a 0.75 b 0.25	a 0.5 b 0.5	a 1 b 0
$0.375 \geq \frac{1}{z}$	P :		a	a	b	

Weighted Pattern Matching

WEIGHTED PATTERN MATCHING

Given a weighted sequence T, a threshold $1/z$, and a solid pattern P, determine all the fragments of T matching P.

$\frac{1}{z} = 0.2$

T :	a 0.25 b 0.75	a 1 b 0	a 0.75 b 0.25	a 0.5 b 0.5	a 1 b 0
P :			a	a	b

Weighted Pattern Matching

Weighted Pattern Matching

Given a weighted sequence T, a threshold $1/z$, and a solid pattern P, determine all the fragments of T matching P.

$\frac{1}{z} = 0.2$	T :	a 0.25 b 0.75	a 1 b 0	a 0.75 b 0.25	a 0.5 b 0.5	a 1 b 0
$0 < \frac{1}{z}$	P :			a	a	b

Weighted Pattern Matching: Results

- n: text length,
- R: text size (represented as a sparse matrix),
- m: pattern length,
- σ: alphabet size,
- $\frac{1}{z}$: threshold.

Weighted Pattern Matching: Results

- n: text length,
- R: text size (represented as a sparse matrix),
- m: pattern length,
- σ: alphabet size,
- $\frac{1}{z}$: threshold.

Christodoulakis et al., 2004 $\mathcal{O}(\sigma n \log m)$ (FFT);

Weighted Pattern Matching: Results

- n: text length,
- R: text size (represented as a sparse matrix),
- m: pattern length,
- σ: alphabet size,
- $\frac{1}{z}$: threshold.

Christodoulakis et al., 2004 $\mathcal{O}(\sigma n \log m)$ (FFT);

Barton et al., 2016 $\mathcal{O}(R \log \min(z, \sigma) + nz)$ (one query to an index);

Weighted Pattern Matching: Results

- n: text length,
- R: text size (represented as a sparse matrix),
- m: pattern length,
- σ: alphabet size,
- $\frac{1}{z}$: threshold.

Christodoulakis et al., 2004 $\mathcal{O}(\sigma n \log m)$ (FFT);

Barton et al., 2016 $\mathcal{O}(R \log \min(z, \sigma) + nz)$ (one query to an index);

this work $\mathcal{O}(R + n \log z)$ (LCP + lookahead scoring).

T :	a 0.25	a 1	a 0.75	a 0.5	a 1	$\frac{1}{z} = 0.2$
	b 0.75	b 0	b 0.25	b 0.5	b 0	

Weighted Pattern Matching: Algorithm

T :	a 0.25 b 0.75	a 1 b 0	a 0.75 b 0.25	a 0.5 b 0.5	a 1 b 0	$\frac{1}{z} = 0.2$
$\mathcal{H}(T)$:	b	a	a	b	a	

Heavy string $\mathcal{H}(T)$: most probable character at each position.

Weighted Pattern Matching: Algorithm

T :	a 0.25 b 0.75	a 1 b 0	a 0.75 b 0.25	a 0.5 b 0.5	a 1 b 0	$\frac{1}{z} = 0.2$
$\mathcal{H}(T)$:	b	a	a	b	a	0.5625
P :	a	a	b			≤ 0.5625

Heavy string $\mathcal{H}(T)$: most probable character at each position.

- Compare P with fragments of $\mathcal{H}(T)$:
  - probability of the fragment is an upper bound.

Weighted Pattern Matching: Algorithm

T :	a 0.25 b 0.75	a 1 b 0	a 0.75 b 0.25	a 0.5 b 0.5	a 1 b 0	$\frac{1}{z} = 0.2$
$\mathcal{H}(T)$:	b	a	a	b	a	0.5625
P :	a	a	b			≤ 0.1875

Heavy string $\mathcal{H}(T)$: most probable character at each position.

- Compare P with fragments of $\mathcal{H}(T)$:
  - probability of the fragment is an upper bound.
- For each mismatch:
  - update the probability bound,
  - break if it drops below threshold.

T :	a 0.25 b 0.75	a 1 b 0	a 0.75 b 0.25	a 0.5 b 0.5	a 1 b 0	$\frac{1}{z} = 0.2$
$\mathcal{H}(T)$:	b	a	a	b	a	0.375
P :		a	a	b		≤ 0.375

Heavy string $\mathcal{H}(T)$: most probable character at each position.

- Compare P with fragments of $\mathcal{H}(T)$:
  - probability of the fragment is an upper bound.
- For each mismatch:
  - update the probability bound,
  - break if it drops below threshold.

Weighted Pattern Matching: Algorithm

T :	a 0.25 b 0.75	a 1 b 0	a 0.75 b 0.25	a 0.5 b 0.5	a 1 b 0	$\frac{1}{z} = 0.2$
$\mathcal{H}(T)$:	b	a	a	b	a	0.375
P :		a	a	b		$= 0.375$

Heavy string $\mathcal{H}(T)$: most probable character at each position.

- Compare P with fragments of $\mathcal{H}(T)$:
  - probability of the fragment is an upper bound.
- For each mismatch:
  - update the probability bound,
  - break if it drops below threshold.
- Use LCE queries in $\mathcal{H}(T)\#P$ to enumerate mismatches.

Weighted Pattern Matching: Algorithm

T :	a 0.25 b 0.75	a 1 b 0	a 0.75 b 0.25	a 0.5 b 0.5	a 1 b 0	$\frac{1}{z} = 0.2$
$\mathcal{H}(T)$:	b	a	a	b	a	0.375
P :			a	a	b	≤ 0.375

Heavy string $\mathcal{H}(T)$: most probable character at each position.

- Compare P with fragments of $\mathcal{H}(T)$:
  - probability of the fragment is an upper bound.
- For each mismatch:
  - update the probability bound,
  - break if it drops below threshold.
- Use LCE queries in $\mathcal{H}(T)\#P$ to enumerate mismatches.

Weighted Pattern Matching: Algorithm

T :	a 0.25 b 0.75	a 1 b 0	a 0.75 b 0.25	a 0.5 b 0.5	a 1 b 0	$\frac{1}{z} = 0.2$
$\mathcal{H}(T)$:	b	a	a	b	a	0.375
P :			a	a	b	≤ 0.375

Heavy string $\mathcal{H}(T)$: most probable character at each position.

- Compare P with fragments of $\mathcal{H}(T)$:
  - probability of the fragment is an upper bound.
- For each mismatch:
  - update the probability bound,
  - break if it drops below threshold.
- Use LCE queries in $\mathcal{H}(T)\#P$ to enumerate mismatches.

T :	a 0.25 b 0.75	a 1 b 0	a 0.75 b 0.25	a 0.5 b 0.5	a 1 b 0	$\frac{1}{z} = 0.2$
$\mathcal{H}(T)$:	b	a	a	b	a	0.375
P :			a	a	b	≤ 0.375

Heavy string $\mathcal{H}(T)$: most probable character at each position.

- Compare P with fragments of $\mathcal{H}(T)$:
  - probability of the fragment is an upper bound.
- For each mismatch:
  - update the probability bound,
  - break if it drops below threshold.
- Use LCE queries in $\mathcal{H}(T)\#P$ to enumerate mismatches.

Weighted Pattern Matching: Algorithm

T :	a 0.25 b 0.75	a 1 b 0	a 0.75 b 0.25	a 0.5 b 0.5	a 1 b 0	$\frac{1}{z} = 0.2$
$\mathcal{H}(T)$:	b	a	a	b	a	0.375
P :			a	a	b	≤ 0

Heavy string $\mathcal{H}(T)$: most probable character at each position.

- Compare P with fragments of $\mathcal{H}(T)$:
  - probability of the fragment is an upper bound.
- For each mismatch:
  - update the probability bound,
  - break if it drops below threshold.
- Use LCE queries in $\mathcal{H}(T)\#P$ to enumerate mismatches.

Weighted Pattern Matching: Algorithm

T :	a 0.25 b 0.75	a 1 b 0	a 0.75 b 0.25	a 0.5 b 0.5	a 1 b 0	$\frac{1}{z} = 0.2$
$\mathcal{H}(T)$:	b	a	a	b	a	
P :			a	a	b	

Heavy string $\mathcal{H}(T)$: most probable character at each position.

- Compare P with fragments of $\mathcal{H}(T)$:
  - probability of the fragment is an upper bound.
- For each mismatch:
  - update the probability bound,
  - break if it drops below threshold.
- Use LCE queries in $\mathcal{H}(T)\#P$ to enumerate mismatches.

Running time analysis:

- at most $\log z$ mismatches per position (before we break);

Weighted Pattern Matching: Algorithm

T :	a 0.25 b 0.75	a 1 b 0	a 0.75 b 0.25	a 0.5 b 0.5	a 1 b 0	$\frac{1}{z} = 0.2$
$\mathcal{H}(T)$:	b	a	a	b	a	
P :			a	a	b	

Heavy string $\mathcal{H}(T)$: most probable character at each position.

- Compare P with fragments of $\mathcal{H}(T)$:
  - probability of the fragment is an upper bound.
- For each mismatch:
  - update the probability bound,
  - break if it drops below threshold.
- Use LCE queries in $\mathcal{H}(T)\#P$ to enumerate mismatches.

Running time analysis:

- at most $\log z$ mismatches per position (before we break);
- $\mathcal{O}(R)$ preprocessing (for $\mathcal{H}(T)$ and the LCE data structure);

Consensus and Generalized Pattern Matching Problems

Weighted Consensus

Given two weighted sequences and a common threshold, decide if there exists a solid string matching both.

Weighted Consensus

Given two weighted sequences and a common threshold, decide if there exists a solid string matching both.

a 0.5 b 0.5	a 0.5 b 0.5	a 1 b 0	a 0 b 1	a 0.5 b 0.5

$\frac{1}{z} = 0.1$

a 0.25 b 0.75	a 1 b 0	a 0.75 b 0.25	a 0.5 b 0.5	a 1 b 0

Weighted Consensus

Given two weighted sequences and a common threshold, decide if there exists a solid string matching both.

a 0.5 b 0.5	a 0.5 b 0.5	a 1 b 0	a 0 b 1	a 0.5 b 0.5	0.125
					$\frac{1}{z} = 0.1$
a 0.25 b 0.75	a 1 b 0	a 0.75 b 0.25	a 0.5 b 0.5	a 1 b 0	0.28125

Weighted Consensus

Given two weighted sequences and a common threshold, decide if there exists a solid string matching both.

a 0.5 b 0.5	a 0.5 b 0.5	a 1 b 0	a 0 b 1	a 0.5 b 0.5	0.125
					$\frac{1}{z} = 0.1$
a 0.25 b 0.75	a 1 b 0	a 0.75 b 0.25	a 0.5 b 0.5	a 1 b 0	0.28125

General Weighted Pattern Matching

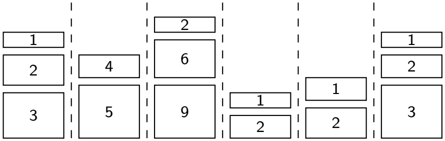
Given an weighted text T, a weighted pattern P, and a threshold $\frac{1}{z}$, decide which fragments of T have a consensus string with P.

Multichoice Knapsack Problem

Multichoice Knapsack

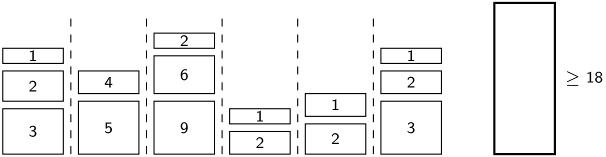
Multichoice Knapsack

- Input:
  - R items in n classes, each item with weight w and value v,



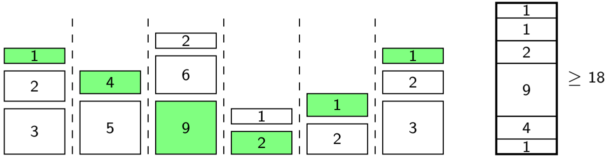
MULTICHOICE KNAPSACK

- Input:
  - R items in n classes, each item with weight w and value v,
  - weight threshold W and value threshold V.



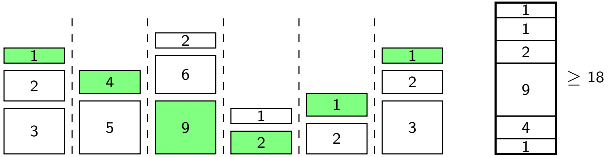
MULTICHOICE KNAPSACK

- Input:
  - R items in n classes, each item with weight w and value v,
  - weight threshold W and value threshold V.
- Output: a choice of 1 item per class, with total weight $\leq W$ and total value $\geq V$.



Multichoice Knapsack

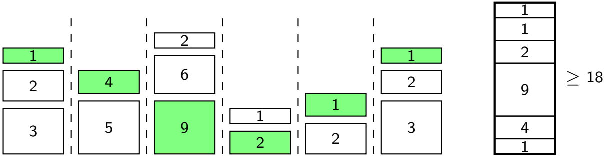
- Input:
  - R items in n classes, each item with weight w and value v,
  - weight threshold W and value threshold V.
- Output: a choice of 1 item per class, with total weight $\leq W$ and total value $\geq V$.



Multichoice Knapsack Problem

Multichoice Knapsack

- Input:
  - R items in n classes, each item with weight w and value v,
  - weight threshold W and value threshold V.
- Output: a choice of 1 item per class, with total weight $\leq W$ and total value $\geq V$.

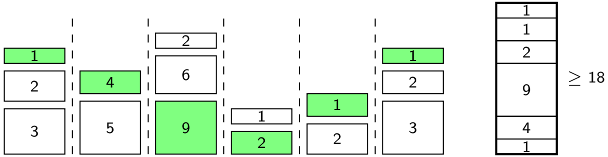


Generalizes Subset Sum and k-Sum.

Multichoice Knapsack Problem

Multichoice Knapsack

- Input:
  - R items in n classes, each item with weight w and value v,
  - weight threshold W and value threshold V.
- Output: a choice of 1 item per class, with total weight $\leq W$ and total value $\geq V$.



Generalizes Subset Sum and k-Sum.

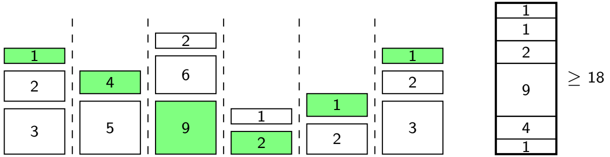
Generalizes Weighted Consensus:

- classes $\leftarrow$ positions,
- items $\leftarrow$ possible characters (at a given position),
- values $\leftarrow$ log-probabilities in the first sequence,
- weights $\leftarrow$ minus log-probabilities in the second sequence.

Multichoice Knapsack Problem

Multichoice Knapsack

- Input:
  - R items in n classes, each item with weight w and value v,
  - weight threshold W and value threshold V.
- Output: a choice of 1 item per class, with total weight $\leq W$ and total value $\geq V$.



Generalizes Subset Sum and k-Sum.

Generalizes Weighted Consensus: (also a converse reduction!)

- classes ← positions,
- items ← possible characters (at a given position),
- values ← log-probabilities in the first sequence,
- weights ← minus log-probabilities in the second sequence.

Results for Weighted Consensus

Upper bounds (algorithms):

Results for Weighted Consensus

Upper bounds (algorithms):

- via meet-in-the-middle for MULTICHOICE KNAPSACK: $\mathcal{O}(n\sigma^{\lceil n/2 \rceil})$ time;

Results for Weighted Consensus

Upper bounds (algorithms):

- via meet-in-the-middle for MULTICHOICE KNAPSACK: $\mathcal{O}(n\sigma^{\lceil n/2 \rceil})$ time;
- our result: $\mathcal{O}(R + \sqrt{z\sigma}(\log\log z + \log\sigma))$ time.

Results for Weighted Consensus

Upper bounds (algorithms):

- via meet-in-the-middle for MULTICHOICE KNAPSACK: $\mathcal{O}(n\sigma^{\lceil n/2 \rceil})$ time;
- our result: $\mathcal{O}(R + \sqrt{z\sigma}(\log\log z + \log\sigma))$ time.

Conditional lower bounds (via MULTICHOICE KNAPSACK):

- no $\mathcal{O}^*(z^{o(1)})$ time unless ETH holds.

Results for Weighted Consensus

Upper bounds (algorithms):

- via meet-in-the-middle for Multichoice Knapsack: $\mathcal{O}(n\sigma^{\lceil n/2 \rceil})$ time;
- our result: $\mathcal{O}(R + \sqrt{z\sigma}(\log\log z + \log\sigma))$ time.

Conditional lower bounds (via Multichoice Knapsack):

- no $\mathcal{O}^*(z^{o(1)})$ time unless ETH holds.
- no $\mathcal{O}^*(z^{1/2-\varepsilon})$ time unless Subset Sum can be improved.

Results for Weighted Consensus

Upper bounds (algorithms):

- via meet-in-the-middle for Multichoice Knapsack: $\mathcal{O}(n\sigma^{\lceil n/2 \rceil})$ time;
- our result: $\mathcal{O}(R + \sqrt{z\sigma}(\log\log z + \log\sigma))$ time.

Conditional lower bounds (via Multichoice Knapsack):

- no $\mathcal{O}^*(z^{o(1)})$ time unless ETH holds.
- no $\mathcal{O}^*(z^{1/2-\varepsilon})$ time unless Subset Sum can be improved.
- no $\tilde{\mathcal{O}}(z^{1/2}\sigma^{1/2-\varepsilon})$ time unless 3-Sum can be improved.

Upper bounds (algorithms):

- via meet-in-the-middle for MULTICHOICE KNAPSACK: $\mathcal{O}(n\sigma^{\lceil n/2 \rceil})$ time;
- our result: $\mathcal{O}(R + \sqrt{z\sigma}(\log\log z + \log\sigma))$ time.
- this talk: $\mathcal{O}(n\sqrt{z\sigma}\log z)$ time.

Conditional lower bounds (via MULTICHOICE KNAPSACK):

- no $\mathcal{O}^*(z^{o(1)})$ time unless ETH holds.
- no $\mathcal{O}^*(z^{1/2-\varepsilon})$ time unless SUBSET SUM can be improved.
- no $\tilde{\mathcal{O}}(z^{1/2}\sigma^{1/2-\varepsilon})$ time unless 3-SUM can be improved.

Meet-in-the-middle Algorithm for Multichoice Knapsack

- Designed by Horowitz & Sahni (1972) for Knapsack.

Meet-in-the-middle Algorithm for Multichoice Knapsack

- Designed by Horowitz & Sahni (1972) for KNAPSACK.
- Running time: $\mathcal{O}(n\sigma^{\lceil n/2 \rceil})$.

Meet-in-the-middle Algorithm for Multichoice Knapsack

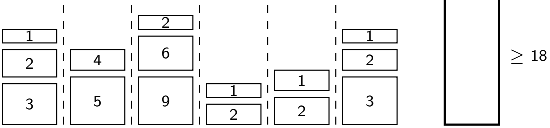
- Designed by Horowitz & Sahni (1972) for KNAPSACK.
- Running time: $\mathcal{O}(n\sigma^{\lceil n/2 \rceil})$.
- Only lower-order-terms improvements
  (for MULTICHOICE KNAPSACK and many special cases).

Meet-in-the-middle Algorithm for Multichoice Knapsack

- Designed by Horowitz & Sahni (1972) for KNAPSACK.
- Running time: $\mathcal{O}(n\sigma^{\lceil n/2 \rceil})$.
- Only lower-order-terms improvements
  (for MULTICHOICE KNAPSACK and many special cases).

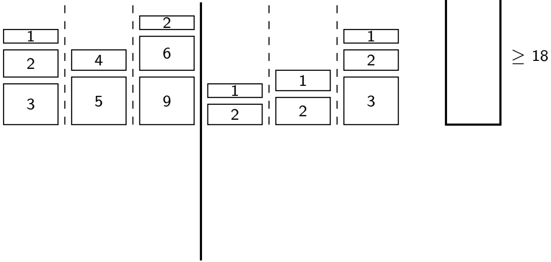
Meet-in-the-middle Algorithm for Multichoice Knapsack

- Designed by Horowitz & Sahni (1972) for KNAPSACK.
- Running time: $\mathcal{O}(n\sigma^{\lceil n/2 \rceil})$.
- Only lower-order-terms improvements
  (for MULTICHOICE KNAPSACK and many special cases).



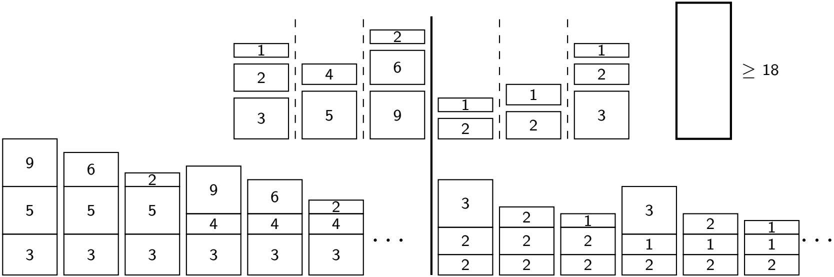
Meet-in-the-middle Algorithm for Multichoice Knapsack

- Designed by Horowitz & Sahni (1972) for KNAPSACK.
- Running time: $\mathcal{O}(n\sigma^{\lceil n/2 \rceil})$.
- Only lower-order-terms improvements
  (for MULTICHOICE KNAPSACK and many special cases).



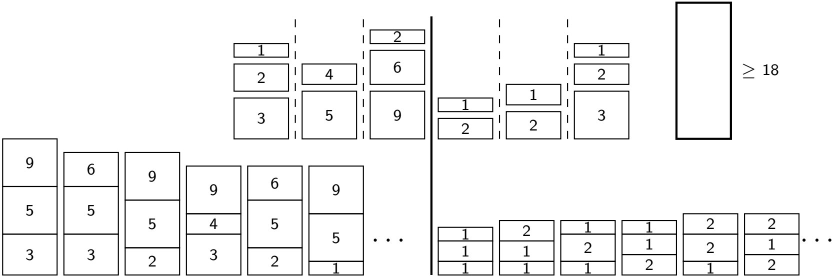
Meet-in-the-middle Algorithm for Multichoice Knapsack

- Designed by Horowitz & Sahni (1972) for KNAPSACK.
- Running time: $\mathcal{O}(n\sigma^{\lceil n/2 \rceil})$.
- Only lower-order-terms improvements (for MULTICHOICE KNAPSACK and many special cases).



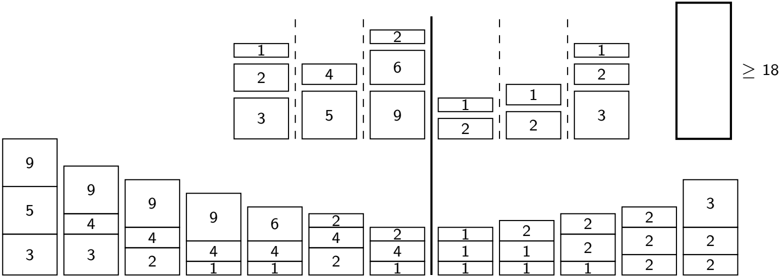
Meet-in-the-middle Algorithm for Multichoice Knapsack

- Designed by Horowitz & Sahni (1972) for KNAPSACK.
- Running time: $\mathcal{O}(n\sigma^{\lceil n/2 \rceil})$.
- Only lower-order-terms improvements
  (for MULTICHOICE KNAPSACK and many special cases).



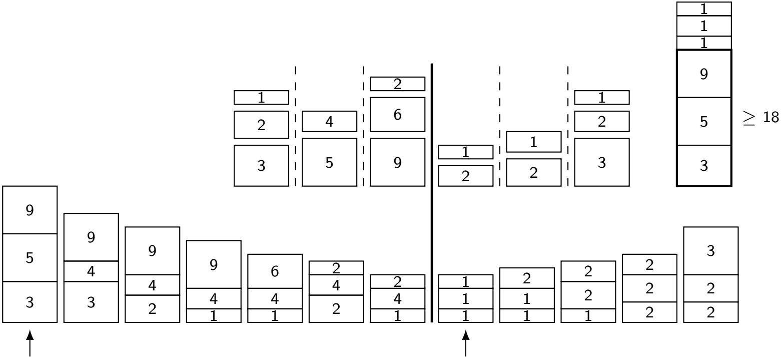
Meet-in-the-middle Algorithm for Multichoice Knapsack

- Designed by Horowitz & Sahni (1972) for KNAPSACK.
- Running time: $\mathcal{O}(n\sigma^{\lceil n/2 \rceil})$.
- Only lower-order-terms improvements
  (for MULTICHOICE KNAPSACK and many special cases).



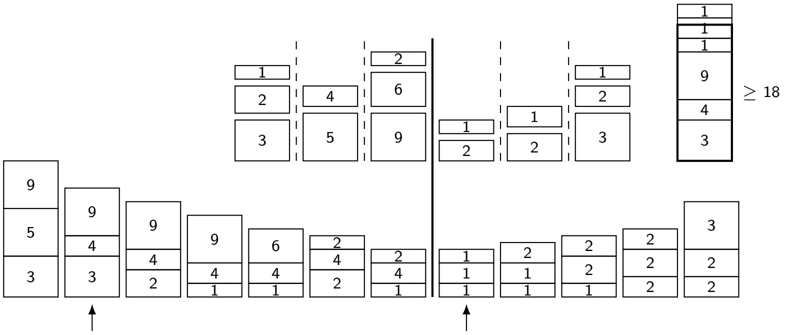
Meet-in-the-middle Algorithm for Multichoice Knapsack

- Designed by Horowitz & Sahni (1972) for Knapsack.
- Running time: $\mathcal{O}(n\sigma^{\lceil n/2 \rceil})$.
- Only lower-order-terms improvements
  (for Multichoice Knapsack and many special cases).



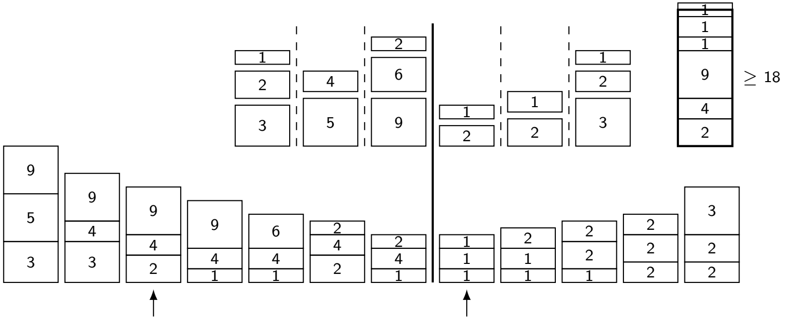
Meet-in-the-middle Algorithm for Multichoice Knapsack

- Designed by Horowitz & Sahni (1972) for KNAPSACK.
- Running time: $\mathcal{O}(n\sigma^{\lceil n/2 \rceil})$.
- Only lower-order-terms improvements
  (for MULTICHOICE KNAPSACK and many special cases).



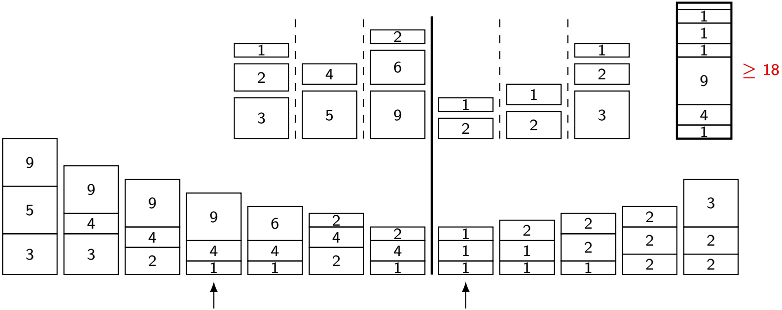
Meet-in-the-middle Algorithm for Multichoice Knapsack

- Designed by Horowitz & Sahni (1972) for KNAPSACK.
- Running time: $\mathcal{O}(n\sigma^{\lceil n/2 \rceil})$.
- Only lower-order-terms improvements
  (for MULTICHOICE KNAPSACK and many special cases).



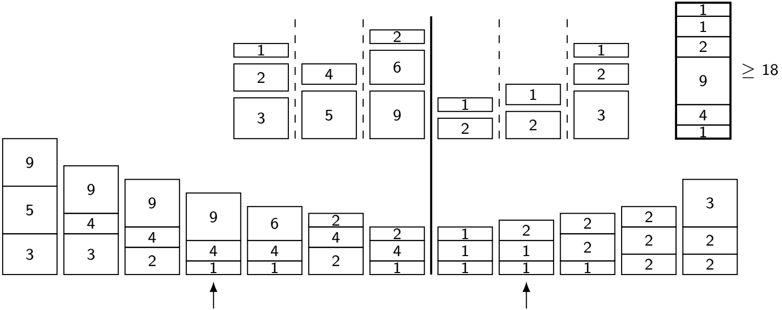
Meet-in-the-middle Algorithm for Multichoice Knapsack

- Designed by Horowitz & Sahni (1972) for KNAPSACK.
- Running time: $\mathcal{O}(n\sigma^{\lceil n/2 \rceil})$.
- Only lower-order-terms improvements
  (for MULTICHOICE KNAPSACK and many special cases).

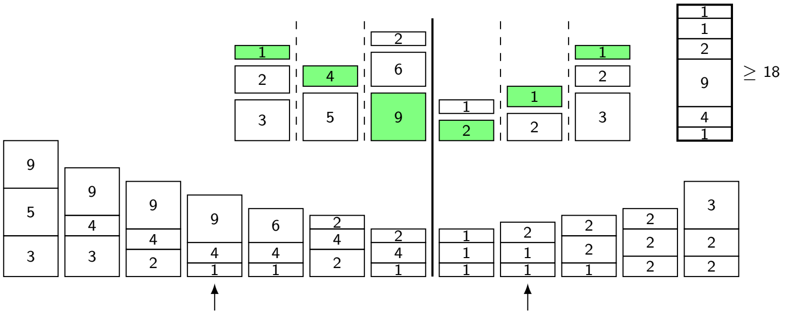


Meet-in-the-middle Algorithm for Multichoice Knapsack

- Designed by Horowitz & Sahni (1972) for KNAPSACK.
- Running time: $\mathcal{O}(n\sigma^{\lceil n/2 \rceil})$.
- Only lower-order-terms improvements (for MULTICHOICE KNAPSACK and many special cases).



- Designed by Horowitz & Sahni (1972) for KNAPSACK.
- Running time: $\mathcal{O}(n\sigma^{\lceil n/2 \rceil})$.
- Only lower-order-terms improvements
  (for MULTICHOICE KNAPSACK and many special cases).



Challenges:

Meet-in-the-middle Algorithm for Weighted Consensus

Challenges:

- How to generate only significant partial solutions?

Meet-in-the-middle Algorithm for Weighted Consensus

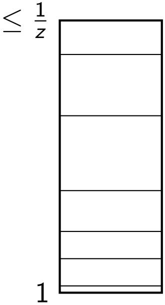
Challenges:

- How to generate only significant partial solutions?
- How to split the domain?

Meet-in-the-middle Algorithm for Weighted Consensus

Challenges:

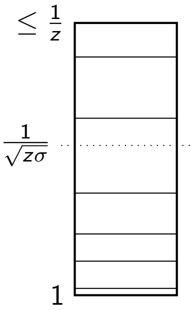
- How to generate only significant partial solutions?
- How to split the domain?



Meet-in-the-middle Algorithm for Weighted Consensus

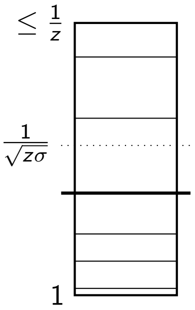
Challenges:

- How to generate only significant partial solutions?
- How to split the domain?



Challenges:

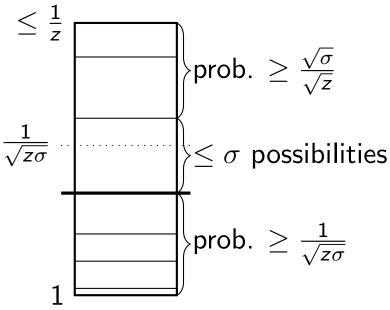
- How to generate only significant partial solutions?
- How to split the domain?



Meet-in-the-middle Algorithm for Weighted Consensus

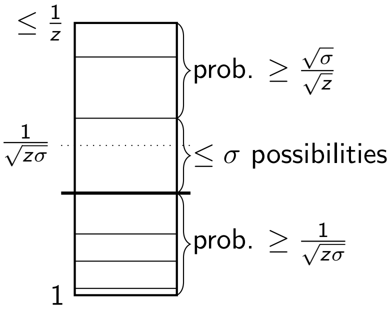
Challenges:

- How to generate only significant partial solutions?
- How to split the domain?



Challenges:

- How to generate only significant partial solutions?
- How to split the domain?



Running time $\mathcal{O}(n\sqrt{z\sigma}\log z)$:

- $\mathcal{O}(n)$ choices for the domain partitioning,
- $\mathcal{O}(\sqrt{z\sigma})$ prefixes and $\mathcal{O}(\frac{\sqrt{z}}{\sqrt{\sigma}} \cdot \sigma)$ suffixes,
- $\mathcal{O}(\log z)$ for sorting.

Summary of Our Further Results

Weighted Consensus

- $\mathcal{O}(R + \sqrt{z\sigma}(\log\log z + \log\sigma))$ time (careful implementation).
  - Preprocessing yields $n = \mathcal{O}(\frac{\log z}{\log \sigma})$.
  - Share some work between partitions considered.

Summary of Our Further Results

Weighted Consensus

- $\mathcal{O}(R + \sqrt{z\sigma}(\log\log z + \log\sigma))$ time (careful implementation).
  - Preprocessing yields $n = \mathcal{O}(\frac{\log z}{\log \sigma})$.
  - Share some work between partitions considered.

Generalized Weighted Pattern Matching

- $\mathcal{O}(n\sqrt{z\sigma}(\log\log z + \log\sigma))$ time.
  - $\leq 2\log z$ mismatches between heavy strings + LCE queries $\rightsquigarrow$ n weighted consensus instances, each for length $\mathcal{O}(\log z)$.

Summary of Our Further Results

Weighted Consensus

- $\mathcal{O}(R + \sqrt{z\sigma}(\log\log z + \log\sigma))$ time (careful implementation).
  - Preprocessing yields $n = \mathcal{O}(\frac{\log z}{\log \sigma})$.
  - Share some work between partitions considered.

Generalized Weighted Pattern Matching

- $\mathcal{O}(n\sqrt{z\sigma}(\log\log z + \log\sigma))$ time.
  - $\leq 2\log z$ mismatches between heavy strings + LCE queries
    $\rightsquigarrow$ n weighted consensus instances, each for length $\mathcal{O}(\log z)$.

Multichoice Knapsack

- Parameter: # of solutions satisfying one of the constraints.
  - Unknown to the algorithm.
- Same ideas, slightly worse lower-order terms.

Summary of Our Further Results

Weighted Consensus

- $\mathcal{O}(R + \sqrt{z\sigma}(\log\log z + \log\sigma))$ time (careful implementation).
  - Preprocessing yields $n = \mathcal{O}(\frac{\log z}{\log \sigma})$.
  - Share some work between partitions considered.

Generalized Weighted Pattern Matching

- $\mathcal{O}(n\sqrt{z\sigma}(\log\log z + \log\sigma))$ time.
  - $\le 2\log z$ mismatches between heavy strings + LCE queries $\rightsquigarrow$ n weighted consensus instances, each for length $\mathcal{O}(\log z)$.

Multichoice Knapsack

- Parameter: # of solutions satisfying one of the constraints.
  - Unknown to the algorithm.
- Same ideas, slightly worse lower-order terms.

Improved Weighted Consensus algorithms for $z = \sigma^{\Theta(1)}$

- Conditionally optimal solutions for $z = \sigma^c$ for any constant c.
- Lower bounds wrt. k-SUM.

Questions?

Thank you for your attention!