The knapsack algorithm in analytical chemistry

Myriam Guillevic, Aurore Guillevic, Martin K. Vollmer, Paul Schlauri, Matthias Hill, Lukas Emmenegger, Stefan Reimann

Laboratory for Air Pollution /Environmental Technology, Empa, Swiss Federal Laboratories for Materials Science and Technology Dübendorf, Switzerland Université de Lorraine, CNRS, Inria, LORIA, Nancy, France

September 23, 2021

https://members.loria.fr/AGuillevic/files/talks/knapsack.py
The software development team, 2018–2021

- Myriam Guillevic, PhD in Climate Sciences, LSCE France & U. of Copenhagen 2019–2021 at Group for Climate Gases, EMPA, ETH, Dübendorf, Switzerland
  https://www.empa.ch/web/s503/climate-gases
  Introduction to algorithms and programming in Java, Python
Aims

**Trace gases:** 0.066% of the (dry) atmosphere, gases other than nitrogen (78.1%), oxygen (20.9%) and argon (0.934%) (i.e. 99.934%, water vapor excluded).

- Measuring trace gases known to be present (e.g. CO$_2$, CH$_4$, CFCl$_3$): **target screening** (mass spectra in databases)
  e.g. banned or regulated substances (Montreal protocol)

- Searching for expected/potential pollutants (known to be used in industry) and start monitoring them before emissions to the atmosphere is rising: **suspect screening** (chemical compound known, e.g. HFO-1234yf)

- Searching for unexpected pollutants or unknown unknowns: **non-target screening** (detecting new substances in the air)
  e.g. industrial disaster (Lubrizol fire in Rouen, France in 2019, Beirut (Beyrouth) harbour explosion in 2020, Leverkusen, Germany, July 27)
How to search for unknown unknowns?
Target screening: Aprecon – GC – ToF-MS

Pre-concentration (APRECON)

Gas Chromatography
GasPro column
Target screening: Aprecon – GC – ToF-MS

Gas Chromatography
GasPro column
Electron impact (EI)
Time-of-Flight
Mass spectrometer
Tofwerk AG
Pre-concentration (APRECON)

EGU 2021 | 2021-04-27 | myriam.guillevic@empa.ch
Target screening: Aprecon – GC – ToF-MS

Electron impact (EI)
Time-of-Flight Mass spectrometer
Tofwerk AG

Pre-concentration (APRECON)

Gas Chromatography GasPro column
EI ToF MS: Electron Ionisation Time-of-Flight Mass Spectrometer

source:
air sample

column:
different molecules
finish at different
retention times (RT)

neutral
fragments
are lost

ionised
fragments
are detected

detector

mass (m/z)
Input data: mass spectrum
Aim: annotate the figure

Intensity (log)

mass (m/z)
Target vs non-target screening

Target screening:

CFC-11
CFCl₃

Instrumental fingerprint

SCS meeting | 2020-08-25 | myriam.guillevic@empa.ch
Target vs non-target screening

Target screening:

CFC-11
CFCl₃

Instrumental fingerprint

Non-target screening

Measured mass [m/z]

Intensity, normalised
Workflow: Knapsack algorithm

Which atoms can be packed together to match the measured masses, ± u?

9 atoms: H, C, N, O, S, F, Cl, Br, I

Weight: 100.936 ± 0.0002 g/mol
(U = 2 ppm)
The knapsack algorithm

Inputs:

• measured mass intervals, e.g.
  \[[34.96751071, 34.97006625],
  [84.92634272, 84.97112964],
  [116.90200574, 116.90847942]\]

• IUPAC masses of atoms →

Run the knapsack on each interval, list all solutions.

<table>
<thead>
<tr>
<th></th>
<th>Mass</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>1.0078250319</td>
</tr>
<tr>
<td>B</td>
<td>11.00930536</td>
</tr>
<tr>
<td>C</td>
<td>12.</td>
</tr>
<tr>
<td>N</td>
<td>14.0030740074</td>
</tr>
<tr>
<td>O</td>
<td>15.9949146223</td>
</tr>
<tr>
<td>F</td>
<td>18.99840316</td>
</tr>
<tr>
<td>P</td>
<td>30.973762</td>
</tr>
<tr>
<td>S</td>
<td>31.97207073</td>
</tr>
<tr>
<td>Cl</td>
<td>34.96885271</td>
</tr>
<tr>
<td>Br</td>
<td>78.9183376</td>
</tr>
<tr>
<td>I</td>
<td>126.9044719</td>
</tr>
</tbody>
</table>
Knapsack Problem in Combinatorics

In a usual context:

- knapsack of limited capacity
- want to maximize the total value of items

Enumeration + dynamic programming
Knapsack Problem in Combinatorics

In a usual context:

- knapsack of limited capacity
- want to maximize the total value of items

Enumeration + dynamic programming

In chemistry:
Mass interval $\leftrightarrow$ knapsack with lower and upper capacity
No optimisation: list all solutions.

$I = [116.90200574, 116.90847942]$ has 2 solutions:
CCl$_3$ mass 116.90655813 \(\in\) I,
H$_{116}$ mass 116.9077037004 \(\in\) I
Knapsack Problem in Combinatorics

In a usual context:

- knapsack of limited capacity
- want to maximize the total value of items

Enumeration + dynamic programming

In chemistry:
Mass interval $\leftrightarrow$ knapsack with lower and upper capacity
No optimisation: list all solutions.
$I = [116.90200574, 116.90847942]$ has 2 solutions:
$\text{CCl}_3$ mass 116.90655813 $\in I$,
$\text{H}_{116}$ mass 116.9077037004 $\in I$

But $\text{H}_{116}$ is not a molecule: Hydrogen has valence 1
$\rightarrow$ remove non-chemical solutions
<table>
<thead>
<tr>
<th>Mass Min</th>
<th>Mass Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>34.96751071</td>
<td>34.97006625</td>
</tr>
<tr>
<td>35.97413780</td>
<td>35.97778836</td>
</tr>
<tr>
<td>36.96406557</td>
<td>36.96750599</td>
</tr>
<tr>
<td>46.96648952</td>
<td>46.97028744</td>
</tr>
<tr>
<td>48.96255817</td>
<td>48.96840119</td>
</tr>
<tr>
<td>59.96022019</td>
<td>59.97131577</td>
</tr>
<tr>
<td>81.93308641</td>
<td>81.93953315</td>
</tr>
<tr>
<td>82.93831759</td>
<td>82.95111397</td>
</tr>
<tr>
<td>83.93024931</td>
<td>83.93724265</td>
</tr>
<tr>
<td>84.92634272</td>
<td>84.97112964</td>
</tr>
<tr>
<td>85.92419323</td>
<td>85.93924313</td>
</tr>
<tr>
<td>97.92364853</td>
<td>97.93896563</td>
</tr>
<tr>
<td>99.90729357</td>
<td>99.94127719</td>
</tr>
<tr>
<td>116.90200574</td>
<td>116.90847942</td>
</tr>
<tr>
<td>117.89455759</td>
<td>117.92205637</td>
</tr>
<tr>
<td>118.89897942</td>
<td>118.90567754</td>
</tr>
<tr>
<td>119.89648859</td>
<td>119.91785117</td>
</tr>
<tr>
<td>120.89523104</td>
<td>120.90302932</td>
</tr>
<tr>
<td>122.88755350</td>
<td>122.90537266</td>
</tr>
<tr>
<td>mass min</td>
<td>mass max</td>
</tr>
<tr>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>34.96751071</td>
<td>34.97006625</td>
</tr>
<tr>
<td>35.97413780</td>
<td>35.97778836</td>
</tr>
<tr>
<td>36.96406557</td>
<td>36.96750599</td>
</tr>
<tr>
<td>46.96648952</td>
<td>46.97028744</td>
</tr>
<tr>
<td>48.96255817</td>
<td>48.96840119</td>
</tr>
<tr>
<td>59.96022019</td>
<td>59.97131577</td>
</tr>
<tr>
<td>81.93308641</td>
<td>81.93953315</td>
</tr>
<tr>
<td>82.93831759</td>
<td>82.95111397</td>
</tr>
<tr>
<td>83.93024931</td>
<td>83.93724265</td>
</tr>
<tr>
<td>84.92634272</td>
<td>84.97112964</td>
</tr>
<tr>
<td>85.92419323</td>
<td>85.93924313</td>
</tr>
<tr>
<td>97.92364853</td>
<td>97.93896563</td>
</tr>
<tr>
<td>99.90729357</td>
<td>99.94127719</td>
</tr>
<tr>
<td>116.90200574</td>
<td>116.90847942</td>
</tr>
<tr>
<td>117.89455759</td>
<td>117.92205637</td>
</tr>
<tr>
<td>118.89897942</td>
<td>118.90567754</td>
</tr>
<tr>
<td>119.89648859</td>
<td>119.91785117</td>
</tr>
<tr>
<td>120.89523104</td>
<td>120.90302932</td>
</tr>
<tr>
<td>122.88755350</td>
<td>122.90537266</td>
</tr>
<tr>
<td>mass min</td>
<td>mass max</td>
</tr>
<tr>
<td>----------</td>
<td>-----------</td>
</tr>
<tr>
<td>34.96751071</td>
<td>34.97006625</td>
</tr>
<tr>
<td>35.97413780</td>
<td>35.97778836</td>
</tr>
<tr>
<td>36.96406557</td>
<td>36.96750599</td>
</tr>
<tr>
<td>46.96648952</td>
<td>46.97028744</td>
</tr>
<tr>
<td>48.96255817</td>
<td>48.96840119</td>
</tr>
<tr>
<td>59.96022019</td>
<td>59.97131577</td>
</tr>
<tr>
<td>81.93308641</td>
<td>81.93953315</td>
</tr>
<tr>
<td>82.93831759</td>
<td>82.95111397</td>
</tr>
<tr>
<td>83.93024931</td>
<td>83.93724265</td>
</tr>
<tr>
<td>84.92634272</td>
<td>84.97112964</td>
</tr>
<tr>
<td>85.92419323</td>
<td>85.93924313</td>
</tr>
<tr>
<td>97.92364853</td>
<td>97.93896563</td>
</tr>
<tr>
<td>99.90729357</td>
<td>99.94127719</td>
</tr>
<tr>
<td>116.90200574</td>
<td>116.90847942</td>
</tr>
<tr>
<td>117.89455759</td>
<td>117.92205637</td>
</tr>
<tr>
<td>118.89897942</td>
<td>118.90567754</td>
</tr>
<tr>
<td>119.89648859</td>
<td>119.91785117</td>
</tr>
<tr>
<td>120.89523104</td>
<td>120.90302932</td>
</tr>
<tr>
<td>122.88755350</td>
<td>122.90537266</td>
</tr>
</tbody>
</table>
The knapsack algorithm in cryptography

Richard Schroeppel and Adi Shamir.

A. M. Odlyzko.
The rise and fall of knapsack cryptosystems.
The knapsack algorithm in computer science

Divide-and-conquer method, target $[m_{\text{min}}, m_{\text{max}}]$: (Thanks to Paul Zimmermann for pointing out the method)

1. Divide the element masses in two balanced sets $A$ and $B$
2. In parallel:
   - List all possible sums in $[0, m_{\text{max}}]$ of masses from set $A$, sort in increasing order $\rightarrow S_A$
   - List all possible sums in $[0, m_{\text{max}}]$ of masses from set $B$, sort in increasing order $\rightarrow S_B$
3. Read onwards the masses from $S_A$ and downwards the masses from $S_B$, find matches $m_{A,i} + m_{B,j} \in [m_{\text{min}}, m_{\text{max}}]$
   - Linear complexity in $\text{length}(S_A) + \text{length}(S_B)$
Our knapsack algorithm in chemistry

Set $A$: masses of multi-valent atoms C (4), N (3), O (2), S (6)
Set $B$: masses of mono-valent atoms H, F, Cl, Br, I

\[
\text{valence}_{\text{max}} = 1 + 0.5 \sum_{e_i \in \{\text{H}, \text{C}, \text{N}, \text{O}, ..., \text{I}\}} n_i(v_i - 2)
\]
Our knapsack algorithm in chemistry

Set $A$: masses of **multi-valent** atoms C (4), N (3), O (2), S (6)
Set $B$: masses of **mono-valent** atoms H, F, Cl, Br, I

$$\text{valence}_{\text{max}} = 1 + 0.5 \sum_{e_i \in \{H,C,N,O,\ldots,I\}} n_i(v_i - 2)$$

1. List all sums in $[0, m_{\text{max}}]$ of masses from set $A$, sort in increasing order $\rightarrow S_A$
2. Compute maximum valence of each solution of $S_A \rightarrow \text{valence}_{\text{max}}$
Our knapsack algorithm in chemistry

Set $A$: masses of **multi-valent** atoms C (4), N (3), O (2), S (6)
Set $B$: masses of **mono-valent** atoms H, F, Cl, Br, I

\[
\text{valence}_{\text{max}} = 1 + 0.5 \sum_{e_i \in \{H, C, N, O, ..., I\}} n_i(v_i - 2)
\]

1. List all sums in $[0, m_{\text{max}}]$ of masses from set $A$, sort in increasing order $\rightarrow S_A$
2. Compute maximum valence of each solution of $S_A \rightarrow \text{valence}_{\text{max}}$
3. List all sums in $[0, m_{\text{max}}]$ of at most $\text{valence}_{\text{max}}$ masses from set $B$, sort in increasing order $\rightarrow S_B$
4. Read onwards the masses from $S_A$ and downwards the masses from $S_B$, find matches $m_{A,i} + m_{B,j} \in [m_{\text{min}}, m_{\text{max}}]$ and check DBE $\geq 0$

Linear complexity in $\text{length}(S_A) + \text{length}(S_B)$

H$_2$  HCl  Cl$_2$  H$_2$O  H$_2$S  NH$_3$  CCl$_4$  CH$_4$
Our knapsack algorithm in chemistry

Observe that:

- atoms in $A$ are heavy (lightest one C of 12.0 $m/z$)
- lightest atom $H \in B$ of mass 1.0078250319, but $\#H$ bounded by valence$_{\text{max}}$
  $\rightarrow$ in average, reduce the length of $S_B$ by a factor 2
SIRIUS software, AGPL https://github.com/boecker-lab/sirius
https://bio.informatik.uni-jena.de
Universität Jena, Germany, Bio-informatic group

Sebastian Böcker and Zsuzsanna Liptak.
A fast and simple algorithm for the money changing problem.

Kai Dührkop, Marcus Ludwig, Marvin Meusel, and Sebastian Böcker.
Faster mass decomposition.
Workflow: mimick fragmentation process

\[ \text{Molecular ion} \]

\[ \text{CCl}_4 \rightarrow \text{e}^- \rightarrow \text{e}^- \rightarrow \text{CCl}_3^+ \rightarrow \text{CCl}_2^+ \rightarrow \text{Cl}^+ \]

Measured fragments
More about graph algorithms: $\text{CCl}_4$

Singleton, Maximal, Node, Leaf.

Fragmentation graph of $\text{CCl}_4$
More about graph algorithms: $\text{CCl}_4$

Singleton, Maximal, Node, Leaf.

Pseudo-fragmentation graph of knapsack fragments w.r.t. partial order

Fragmentation graph of $\text{CCl}_4$
More about graph algorithms: CCl₄

Output of knapsack: a list of candidate fragment formulas
Define a partial order on the fragment formulas
Cl \leq CCl \leq CCl₂ \leq CCl₃ \leq HCCl₃, but COS incomparable
Build a Directed Acyclic Graph according to the ordering
Setting edges has quadratic complexity in the number of vertices (nodes)
More about graph algorithms: $\text{CCl}_4$

Output of knapsack: a list of candidate fragment formulas
Define a partial order on the fragment formulas
$\text{Cl} \leq \text{CCl} \leq \text{CCl}_2 \leq \text{CCl}_3 \leq \text{HCCl}_3$, but COS incomparable

Build a Directed Acyclic Graph according to the ordering
Setting edges has quadratic complexity in the number of vertices (nodes)

- Fragments in batches, per target interval mass, decreasing mass
- All fragments in one batch are incomparable (mass diff $\ll 1$)
More about graph algorithms: CCl$_4$

Output of knapsack: a list of candidate fragment formulas
Define a partial order on the fragment formulas
Cl $\leq$ CCl $\leq$ CCl$_2$ $\leq$ CCl$_3$ $\leq$ HCCl$_3$, but COS incomparable
Build a Directed Acyclic Graph according to the ordering
Setting edges has quadratic complexity in the number of vertices (nodes)
- Fragments in batches, per target interval mass, decreasing mass
- All fragments in one batch are incomparable (mass diff $\ll 1$)

Much faster complexity, starting with the heaviest target mass
1. First batch: roots (maximal elements)
More about graph algorithms: CCl₄

Output of knapsack: a list of candidate fragment formulas
Define a partial order on the fragment formulas
CCl ≤ CCl₂ ≤ CCl₃ ≤ HCCl₃, but COS incomparable
Build a Directed Acyclic Graph according to the ordering
Setting edges has quadratic complexity in the number of vertices (nodes)
  • Fragments in batches, per target interval mass, decreasing mass
  • All fragments in one batch are incomparable (mass diff ≪ 1)

Much faster complexity, starting with the heaviest target mass

1. First batch: roots (maximal elements)
2. Process one batch of fragments as a whole
   2.1 Compare each fragment to the roots
   2.2 If a subfragment, recursively visit the children until it is a leaf, set a new edge from the parent node
   2.3 if incomparable to any root, keep it aside
More about graph algorithms: CCl$_4$

Output of knapsack: a list of candidate fragment formulas
Define a partial order on the fragment formulas
Cl \leq CCl \leq CCl2 \leq CCl3 \leq HCCl3, but COS incomparable
Build a Directed Acyclic Graph according to the ordering
Setting edges has quadratic complexity in the number of vertices (nodes)

- Fragments in batches, per target interval mass, decreasing mass
- All fragments in one batch are incomparable (mass diff \ll 1)

Much faster complexity, starting with the heaviest target mass

1. First batch: roots (maximal elements)
2. Process one batch of fragments as a whole
   2.1 Compare each fragment to the roots
   2.2 If a subfragment, recursively visit the children until it is a leaf, set a new edge from the parent node
   2.3 if incomparable to any root, keep it aside
3. update the list of roots with the incomparable fragments of the batch
More about graph algorithms: $\text{CCl}_4$

Singleton, Maximal, Node, Leaf.
More about graph algorithms: $\text{CCl}_4$

- Singleton
- Maximal
- Node
- Leaf

$\text{CSBr}$, $\text{C}_2\text{S}_3$
More about graph algorithms: $\text{CCl}_4$

- Singleton, Maximal, Node, Leaf.

- $\text{CSBr}$, $\text{C}_2\text{S}_3$, $\text{HCCl}_3$

- $\text{OSCl}_2$

- $\text{FS}_2\text{Cl}$
More about graph algorithms: $\text{CCl}_4$

Singleton, Maximal, Node, Leaf.

- CSBr
- $\text{C}_2\text{S}_3$
- HCCl$_3$
- OSiCl$_2$
- HCCl$_3$
- CCl$_3$
- FS$_2$Cl
More about graph algorithms: $\text{CCl}_4$

Singleton, Maximal, Node, Leaf.

Graph nodes:
- $\text{CSBr}$
- $\text{C}_2\text{S}_3$
- $\text{HCCl}_3$
- $\text{OSCl}_2$
- $\text{NOCl}_2$
- $\text{CCl}_3$
- $\text{HS}_2\text{Cl}$
- $\text{HO}_2\text{SCI}$
- $\text{FS}_2\text{Cl}$
More about graph algorithms: $\text{CCl}_4$

Singleton, Maximal, Node, Leaf.

$\text{OSCl}_2$, $\text{NOCl}_2$, $\text{COCl}_2$, $\text{CCl}_3$, $\text{HCCl}_3$, $\text{H}_2\text{S}_3$, $\text{HS}_2\text{Cl}$, $\text{HO}_2\text{SCI}$, $\text{FS}_2\text{Cl}$
More about graph algorithms: CCl₄

Singleton, Maximal, Node, Leaf.
More about graph algorithms: \( \text{CCl}_4 \)

**Singleton, Maximal, Node, Leaf.**

Diagram with various chemical compounds and their connections, including:
- \( \text{CSBr} \)
- \( \text{C}_2\text{S}_3 \)
- \( \text{HCCl}_3 \)
- \( \text{H}_2\text{S}_3 \)
- \( \text{OSCl}_2 \)
- \( \text{NOCl}_2 \)
- \( \text{COCl}_2 \)
- \( \text{CCl}_3 \)
- \( \text{HS}_2\text{Cl} \)
- \( \text{HO}_2\text{SCl} \)
- \( \text{H}_2\text{OSCl} \)
- \( \text{HNCl}_2 \)
- \( \text{H}_2\text{FS}_2 \)
- \( \text{FS}_2\text{Cl} \)
More about graph algorithms: $\text{CCl}_4$

- Singleton
- Maximal
- Node
- Leaf.
More about graph algorithms: $\text{CCl}_4$

Singleton, Maximal, Node, Leaf.
More about graph algorithms: $\text{CCl}_4$

- Singleton
- Maximal
- Node
- Leaf

Graph diagram with nodes and edges representing chemical structures.
More about graph algorithms: $\text{CCl}_4$

Singleton, Maximal, Node, Leaf.
More about graph algorithms: $\text{CCl}_4$

- Singleton
- Maximal
- Node
- Leaf

Diagram with nodes labeled with chemical compounds such as $\text{CSBr}$, $\text{C}_2\text{S}_3$, $\text{HCCl}_3$, $\text{H}_2\text{S}_3$, $\text{COS}$, $\text{OSCl}_2$, $\text{NOCl}_2$, $\text{COCl}_2$, $\text{CCl}_3$, $\text{HCCl}_2$, $\text{HS}_2\text{Cl}$, $\text{HO}_2\text{SCI}$, $\text{H}_2\text{OSCI}$, $\text{HNCl}_2$, $\text{H}_2\text{FS}_2$, $\text{FS}_2\text{Cl}$, $\text{HCl}$, $\text{H}_2\text{FS}_2$, $\text{FS}_2\text{Cl}$, and $\text{FS}_2$. The nodes are connected to represent graph algorithms.
More about graph algorithms: $\text{CCl}_4$

- **Singleton**, **Maximal**, **Node**, **Leaf**.

![Diagram](image-url)
More about graph algorithms: \( \text{CCl}_4 \)

- **Singleton**
- **Maximal**
- **Node**
- **Leaf**

Remove singletons
More about graph algorithms: $\text{CCl}_4$

- Singleton, Maximal, Node, Leaf.

Remove singletons
Optimise isotopic profiles

Aim: eliminate all unlikely candidate formulas.

1. Compute the intensity profile of isotopocules →
2. Define a likelihood estimator
3. Fit the theoretic intensities to the measured signal (next slide)
4. Update pseudo-frag graph:
   Remove candidate formulas below LOD (limit of detection)
   Remove new singletons
Workflow: optimise isotopic ‘profiles’

CCl$_2^+$

HCCl$_2^+$

OCl$_2^+$

Isotopologue profiles

Python

lmfit

· $k_1$

· $k_2$

· $k_3$
Workflow: optimise isotopic ‘profiles’

knapsack

Isotopologue profiles

Python lmfit

· $k_1$

· $k_2$

· $k_3$

Optimised contributions

Measured limit of detection

Nachweisgrenze

$\text{CCL}_2^+$

$\text{HCCl}_2^+$

$\text{OCl}_2^+$
Workflow: optimise isotopic ‘profiles’

knapsack

Isotopologue profiles

Python lmfit

· $k_1$

· $k_2$

· $k_3$

Optimised contributions

Measured limit of detection
Nachweisgrenze

$\text{CCl}_2^+$

$\text{HCCl}_2^+$

$\text{OCl}_2^+$
Results – example for CCl$_4$

- 19 measured masses
- 23 knapsack solutions using C, H, N, O, S, F, Cl, Br, I
- 3 singletons removed
- 2 solutions < LOD
- 98% correctly assigned signal
- Runtime: 4 s on a laptop
Result: final graph for CCl$_4$
Result: final graph for CCl₄

Molecular Ion CCl₄ not in the graph!!!
In 40% of measured samples, the molecular ion is not measured (due to the Electron Ionisation technique)
Field work at Jungfraujoch, 3500m

https://www.empa.ch/web/s604/jungfraujoch-bauarbeiten
Validation of the results

More than 75 newly identified substances in Dübendorf air.

1. Buy the suspected substances on catalog
2. Measure with the same machine and same settings
3. Check Retention Time (RT), and mass spectra: peaks at same masses, same proportion of peak intensities

24 substances validated so far.

Issues:

• unavailable substances (exist, but cannot buy them on catalog)
• too toxic for shipping
• too costly ($1000/5g) (our threshold cost: $750/5g)
• all are banned substances
• special authorization from Ministry of Environment for customs department
Future Work

Solve the knapsack problem with LLL?

- François Morain says it will work
- Paul Zimmermann says it will work
- Léo Ducas says it will work

It's only a question of time and human resources...
Conclusion

Fruitful collaboration between computer-scientist and environmental science researchers and engineers

- learned about chemistry
- learned about how to teach CS to senior researchers and engineers
- co-authors learned about algorithms, Python programming, software architecture, and development tools (git)

Python source code released on June 30, 2021 with LGPL license at https://gitlab.inria.fr/guillevi/alpinac/
Paper accepted at the Journal of Cheminformatics
Preprint at https://hal.inria.fr/hal-03176025

EMPA will be looking to hire a computer scientist.