SUBSTRUCTURE SEARCHING FACE-OFF

John May and Roger Sayle
NextMove Software
Cambridge, UK
Molecular Identity
is foo the same as bar?

Substructure
is foo part of bar?

Chemical Similarity
how much is foo like bar?

“A grand unified theory: the three fundamental forces of Cheminformatics” - R Sayle et al.
SUBSTRUCTURE SEARCH

Some of the many uses:

- Fragment-based lead discovery
- Murcko Scaffolds
- Matched pairs
- Functional groups, atom types
- Similarity Keys (MACCS/CACTVS)
- Toxicology - Tim Allen’s MIE models

Substructure searches are infrequent*...
...they can be slow - correlation/causation?

*~10x more unique similarity queries in Structure Query Collection
ANATOMY OF A SUBSEARCH

1. Generate fingerprint

- **CBO query**
- **query-fp**
- **fp type**
- **fp index type**
  - none
  - linear
  - sub-linear

2. Fingerprint Scan

- **CCO**
- **NCC**
- **CCCCO**
- **CCl**
- **CBr**
- **C(=O)O**

- **mol format**
  - line-notation
  - binary

3. Atom-match

- **ff type**
- **VF2**
- **Ullmann**

- **storage**
  - flatfile
  - RDMS

*CBr is actually filtered with this fingerprint but left to show the FP is generally not perfect (precision<1)
WHY CAN IT FEEL SLOW?

Identity and Similarity Search
- different queries, similar amount of work

Substructure Search
- different queries, potentially huge difference in work

Rough est. for ~10 mil compounds
- Identity, index canonical form - b-tree: $\leq 1$ ms
- Similarity, index fingerprint - linear scan: $\leq 100$ ms
- Substructure, index fingerprint + structure
  - linear FP scan, atom-based match: $\geq 100$ ms
  - sub-linear FP scan, atom-based match: $\geq 5$ ms
STRATEGIES?

Parallelisation
resource contention from multiple users
Limit query count (first 10)
must decide which ones are first
Limit time (60 s in eMolecules/SureChEMBL)
hits depend on hardware/workload
Cache or filter pathological queries
how to predict?
Limit screen count (fastsearch)
may get no hits?
EXISTING BENCHMARKS

Several tools publish isolated performance figures and should be commended.

“what performance can you expect”

Alas different hardware, queries, and target dataset limit meaningful conclusions and comparisons.

Using the same hardware, queries, and target dataset we present the execution efficiency (how fast). Retrieval efficiency (hits) will be touched upon.
Our Benchmark:

Count Total Number of Hits
 Needed moderate size collection
  ChEMBL ~1.5 mil, too easy, fits easily in a memory cache
  PubChem Compound ~70 mil, too large for many tools

eMolecules 6,996,230 compounds
  performed minimal cleanup
  eMol IDs will be made available
Structure Query Collection (SQC)

3,488 BindingDB_substructure queries

3,342 connected, 121 fragments

User generated
Includes pathological queries

Excluded fragments - some consider benzene as bad:
- c1ccccc1.c1ccccc1 slow
- c1ccccc1.c1ccccc1.c1ccccc1 very slow
- c1ccccc1.c1ccccc1.c1ccccc1.c1ccccc1.c1ccccc1 ouch

Andrew Dalke's BitBucket Project - https://bitbucket.org/dalke/sqc
Duplicates due to **aromaticity** or **hydrogens**

- \([H]C1=CN=CN=C1[H]\) pyrimidine
- \([H]C1=CN=CN=C1\) pyrimidine
- \(C1=CN=CN=C1\) pyrimidine
- \(C1=CC=CC=C1\) benzene
- \(c1cccccc1\) benzene

**3,140/3,342** unique - but used the non-unique set:

- Hydrogens change semantics of match
- Queries run unmodified but standardised* to **queries-clean** when needed, tools that expect “SMARTS”

*aromatised, suppressed-H, atom-stereo removed
## Tools Evaluated

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Chemistry</th>
<th>Storage</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>arthor</td>
<td>standalone</td>
<td>NextMove</td>
<td>Flatfile</td>
<td>Serialised</td>
</tr>
<tr>
<td>bingo-pgsql</td>
<td>cartridge</td>
<td>Indigo</td>
<td>PostgreSQL</td>
<td>Serialised</td>
</tr>
<tr>
<td>bingo-orcl</td>
<td>cartridge</td>
<td>Indigo</td>
<td>Oracle</td>
<td>Serialised</td>
</tr>
<tr>
<td>bingo-nosql</td>
<td>standalone</td>
<td>Indigo</td>
<td>Flatfile</td>
<td>Serialised</td>
</tr>
<tr>
<td>fastsearch</td>
<td>standalone</td>
<td>Open Babel</td>
<td>Flatfile</td>
<td><strong>SMILES</strong>¹</td>
</tr>
<tr>
<td>jcart</td>
<td>standalone/cartridge</td>
<td>ChemAxon</td>
<td>Oracle</td>
<td>Serialised</td>
</tr>
<tr>
<td>jcart-st (1 thread)</td>
<td>standalone/cartridge</td>
<td>ChemAxon</td>
<td>Oracle</td>
<td>Serialised</td>
</tr>
<tr>
<td>mychem</td>
<td>cartridge</td>
<td>Open Babel</td>
<td><strong>MySQL</strong></td>
<td>Serialised</td>
</tr>
<tr>
<td>pgchem</td>
<td>cartridge</td>
<td>Open Babel</td>
<td>PostgreSQL</td>
<td>Serialised</td>
</tr>
<tr>
<td>orchem</td>
<td>cartridge</td>
<td>CDK</td>
<td>Oracle</td>
<td>Serialised²</td>
</tr>
<tr>
<td>rdcart</td>
<td>cartridge</td>
<td>RDKit</td>
<td>PostgreSQL</td>
<td>Serialised</td>
</tr>
<tr>
<td>rdluc</td>
<td>standalone</td>
<td>RDKit</td>
<td>Lucene</td>
<td><strong>SMILES</strong></td>
</tr>
<tr>
<td>rdsmsgrep</td>
<td>standalone</td>
<td>RDKit</td>
<td>SMILES</td>
<td><strong>SMILES</strong></td>
</tr>
<tr>
<td>tripod-ss</td>
<td>standalone</td>
<td>ChemAxon</td>
<td>Oracle Cache</td>
<td><strong>SMILES</strong>³</td>
</tr>
</tbody>
</table>

¹) FastSearch format depends on input. 2) OrChem serialises as a string rather than binary. 3) Tripod’s Search Search uses molfiles by default. **Underlined:** queries-clean used
# MORE CARTRIDGES

<table>
<thead>
<tr>
<th>NAME</th>
<th>Type</th>
<th>Chemistry</th>
<th>Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pinpoint</td>
<td>Cartridge</td>
<td>Dotmatics</td>
<td>Oracle</td>
</tr>
<tr>
<td>Direct</td>
<td>Cartridge</td>
<td>BIOVIA/Accelrys/MDL</td>
<td>Oracle</td>
</tr>
<tr>
<td>Accord</td>
<td>Cartridge</td>
<td>BIOVIA/Accelrys</td>
<td>Oracle</td>
</tr>
<tr>
<td>ABCD</td>
<td>Cartridge</td>
<td>J&amp;J</td>
<td>Oracle</td>
</tr>
<tr>
<td>DayCart</td>
<td>Cartridge</td>
<td>Daylight</td>
<td>Oracle/Postgres</td>
</tr>
<tr>
<td>Torus</td>
<td>Cartridge</td>
<td>Digital Chem</td>
<td>Oracle</td>
</tr>
<tr>
<td>OpenEye Cartridge</td>
<td>Cartridge</td>
<td>OpenEye</td>
<td>Oracle</td>
</tr>
<tr>
<td>MolCart</td>
<td>Cartridge</td>
<td>ICM</td>
<td>MySQL</td>
</tr>
<tr>
<td>MolSQL</td>
<td>Cartridge</td>
<td>Scilligence</td>
<td>SQL Server</td>
</tr>
<tr>
<td>Chord</td>
<td>Cartridge</td>
<td>OpenEye</td>
<td>PostgreSQL</td>
</tr>
<tr>
<td>OpenChord</td>
<td>Cartridge</td>
<td>Open Babel / RDKit</td>
<td>PostgreSQL</td>
</tr>
<tr>
<td>IC Cartridge</td>
<td>Cartridge</td>
<td>InfoChem</td>
<td>Oracle</td>
</tr>
<tr>
<td>AUSPYX</td>
<td>Cartridge</td>
<td>Tripos</td>
<td>Oracle</td>
</tr>
<tr>
<td>Oracle Cartridge</td>
<td>Cartridge</td>
<td>PerkinElemer</td>
<td>Oracle</td>
</tr>
</tbody>
</table>

Some not externally available (ABCD) or being retired (Accord)
### MORE STANDALONES

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>CHEMISTRY</th>
<th>STORAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChemFP</td>
<td>standalone</td>
<td>Multiple</td>
<td>FPS/BFPS in-memory</td>
</tr>
<tr>
<td>Similr</td>
<td>standalone</td>
<td>?</td>
<td>MongoDB</td>
</tr>
<tr>
<td>Data Warrior</td>
<td>standalone</td>
<td>Acetlion</td>
<td>In-memory</td>
</tr>
<tr>
<td>Ambit - OpenTox</td>
<td>standalone</td>
<td>CDK/Ambit</td>
<td>MySQL</td>
</tr>
<tr>
<td>MolecularLucene</td>
<td>standalone</td>
<td>CDK</td>
<td>Lucene</td>
</tr>
<tr>
<td>Wikipedia CSE</td>
<td>standalone</td>
<td>Acetlion (JS)</td>
<td>In-browser-memory</td>
</tr>
<tr>
<td>MOLDB6</td>
<td>standalone</td>
<td>checkmol/matchmol</td>
<td>MySQL</td>
</tr>
<tr>
<td>ChemSearch(^1)</td>
<td>SQL</td>
<td>SQL</td>
<td>Oracle/PostgreSQL</td>
</tr>
<tr>
<td>SQLMOL</td>
<td>SQL</td>
<td>SQL</td>
<td>MS SQL</td>
</tr>
</tbody>
</table>

**ChemFP, Similr, MolecularLucene**
- currently **similarity** only

**MOLDB6**
- index time for eMol est. at **22** days... we started last week

---

1: Golovin A and Henrick K. Chemical Substructure Search in SQL. *JCIM*. 2008. 49(1)
CHANGES FROM “OUT OF THE BOX”

MyChem and RDKit Lucene
- fingerprint screen workaround
- OB FP2 and Avalon FP, no bits ≠ no hits

Tripod Search Server
- Load from SMILES instead of molfile
- Avoid aromaticity recalculation
- Used larger cache size
MEASUREMENT IS NOT PROPHECY

Large number of variables

Time taken for this benchmark, with these tool versions, on a machine, with this load. Tools were setup as per user manuals (shared memory etc).

A “normal” desktop, £££ less than iPhone 6+.

CentOS 7

Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz
16 GB RAM

“Trust has no place in science” - NextMove Software will make inputs available for others to test

1: http://benchmarksgame.alioth.debian.org/
3342 QUERIES
6996230 TARGETS
≤23,381,400,660 MATCHES LATER
19 queries skipped in one or more tools
3,323/3,342 that were executed by all

2 Non-terminal hydrogens
C1N[H]0C[H]1

1 Invalid stereochemistry
CC[C@@H]

10 Atom classes
C1C[N:1]CNC1

6 Bond stereo not-implemented (should have cleaned)
“We demand rigidly defined areas of doubt and uncertainty!”

o-xylene
RETRIEVAL DIFFERENCES

“We demand rigidly defined areas of doubt and uncertainty!”

Somewhere between 440,901 and 529,386

- rdcart 500,126
- rdluc 500,053
- bingo (pgsql/orcl) 500,865
- bingo (nosql) 498,427
- tripod-ss 509,655
- jcart 444,541
- fastsearch 440,901
- orchem 529,386
- arthor 444,553

![o-xylene molecule]
many queries, running **quick**

Where the line drops from 3323, that is the fastest query. Where it falls below 0, that was the slowest.

Area under curve ~ total elapsed time for all queries. However log, log plot so can be deceptive.
No FP screen, all queries take around 5 mins for `rdsmigrep` (no sanitise). `rdcart` slowest query is nearly as slow.
rdcart worse case is only better due to mol serialisation. rdluc / rdsmigrep both use SMILES.

FP scan degraded performance for these queries
Bingo Oracle and PostgreSQL perform similarly. Small difference is likely measurement inaccuracies.
The recent “NoSQL” (flatfile) version is about twice as fast - DB overhead?
jcart was the fastest cartridge. By default it uses all available processors...
...limiting to a single thread (st) we observed a similar performance to the Bingo cartridges.
pgchem and mychem have a different shape curve. This turns out to be hydrogen sprouting for some queries.

fastsearch would also show this but got queries-clean
Both mychem and fastsearch do a linear FP scan. mychem does this via a SQL statement, fastsearch does it sequentially from the file.
**pghcem** and **rdcart** both use a GIST, r-tree based index. This provides sub-linear screening for some queries.

The curves are similar and we would therefore expect **pghcem** to also be ~ 6 hrs with the hydrogen issues resolved.
The curve is shallower because `pgchem` and `mychem` use serialised molecules.
tripod-ss use the JChem atom-match and path fingerprints. jcart was fastest on overall time...

...but sub-linear screens clearly win for many easy queries. Notice jcart ran ~1000 queries faster.
Unfortunately **orchem** was the slowest cartridge measured. The slowest queries were similar to others using line-notations.

Recent CDK improvements could make it faster.
Comparing *queries-clean* to default query set.
OTHER CONSIDERATIONS

What are the hits? o-xylene example

How do they scale?

- Index, tree-based indexes will have many more internal nodes
- Memory usage
  - binary < SMILES < molfile
  - fingerprint length

PubChem Compound currently ~10x eMolecules
BENCHMARK SUMMARY

Many tools have reasonable performance for most queries but are hindered by their worst cases. Minimal difference between Oracle vs PostgreSQL

OrChem vs JCart (slowest and fastest cartridges)

MyChem and pgchem will benefit from improved query preparation (hydrogen handling)

Domain specific indexes preferable to table selects.

Efficient IO

Image: http://timvdm.blogspot.co.uk/
Much work has been done on optimising fingerprints. Screen performance measured for several of the tools:

- only partially explains observed time differences
- occasional v. bad precision for some queries
- hydrogens are difficult to encode

**Maybe False Negatives! - but they’re relative**

<table>
<thead>
<tr>
<th>TOOL</th>
<th>TYPE</th>
<th>SCREENOUT</th>
<th>PRECISION</th>
</tr>
</thead>
<tbody>
<tr>
<td>bingo</td>
<td>Hashed Trees/Rings</td>
<td>98.63%</td>
<td>80.9%</td>
</tr>
<tr>
<td>rdcart</td>
<td>Hashed Patterns</td>
<td>97.95%</td>
<td>59.9%</td>
</tr>
<tr>
<td>jcart,tripod-ss</td>
<td>Hashed Paths</td>
<td>98.15%</td>
<td>59.7%</td>
</tr>
<tr>
<td>pgchem,mychem,fastsearch</td>
<td>Hashed Paths/Rings</td>
<td>98.2%</td>
<td>57.3%</td>
</tr>
</tbody>
</table>

1. [http://nextmovesoftware.com/blog/2015/02/16/](http://nextmovesoftware.com/blog/2015/02/16/)
AN EFFICIENT ATOM-MATCH

• Query optimisation
• Database preprocessing
• Efficient binary representation
  • eMolecules, 1.2GiB
• Matching done on representation
FINGERPRINTING MATERS

Arthor - 0% screen-out
23,290,449,670 atom-matches

Bingo - 98.63% screen-out
319,079,160 atom-matches

Can test how much different fingerprints matter...
Provide arthor a pre-computed list of screen hits
Times therefore presume instant bit screen
False negatives are relative to 0% screen-out.
# Fingerprinting Matters

<table>
<thead>
<tr>
<th>Fingerprint</th>
<th>Length</th>
<th>Density</th>
<th>Time</th>
<th>False Negatives</th>
</tr>
</thead>
<tbody>
<tr>
<td>arthor</td>
<td>-</td>
<td>-</td>
<td>27m35s</td>
<td>-</td>
</tr>
<tr>
<td>+rdpath</td>
<td>2048</td>
<td>61.90%</td>
<td>2m52s</td>
<td>61,915 (0.02%)</td>
</tr>
<tr>
<td>+maccs</td>
<td>166</td>
<td>29.08%</td>
<td>2m26s</td>
<td>140,968,649 (57.63%)</td>
</tr>
<tr>
<td>+dalke353</td>
<td>353</td>
<td>6.59%</td>
<td>2m2s</td>
<td>0</td>
</tr>
<tr>
<td>+obfp2</td>
<td>1021</td>
<td>12.55%</td>
<td>1m32s</td>
<td>641,712 (0.26%)</td>
</tr>
<tr>
<td>+jpath</td>
<td>1024</td>
<td>23.36%</td>
<td>1m7s</td>
<td>41,889 (0.01)</td>
</tr>
<tr>
<td>+indigo-sub</td>
<td>1600</td>
<td>23.09%</td>
<td>55s</td>
<td>128,288 (0.05%)</td>
</tr>
<tr>
<td>+indigo-sub+ext</td>
<td>1624</td>
<td>22.48%</td>
<td>54s</td>
<td>128,288 (0.05%)</td>
</tr>
</tbody>
</table>
n.b. < 3323 because some queries had no hits from the fingerprint screen
CONCLUSIONS

Fast searching depends on many factors

• index data-structure
• fingerprint type
• molecule representation
• atom-match algorithm

Fingerprint optimisation does mater but is not the performance bottleneck and can’t always help:

Generic patterns, a–a
Hydrogens, [H]N([H])C
ACKNOWLEDGEMENTS

Daniel Lowe and Noel O’Boyle, NextMove Software

Andrew Dalke
- assembling Structure Query Collection (SQC)

Michael Gilson and Tiqing Liu
- providing BindingDB (http://www.bindingdb.org/)
  queries for SQC

Tool and cartridge developers
Further optimisation and keeping file open: 36 s including inverted index FP screen, single threaded:

<table>
<thead>
<tr>
<th>Time (ms)</th>
<th>Queries Not Finished (n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1s</td>
<td>3323</td>
</tr>
<tr>
<td>10s</td>
<td>99%</td>
</tr>
<tr>
<td>30s</td>
<td>90%</td>
</tr>
<tr>
<td>1m</td>
<td>10%</td>
</tr>
<tr>
<td>5m</td>
<td>1%</td>
</tr>
</tbody>
</table>
SOME FALSE NEGATIVES

Most false negatives are due to Kekulisation/Aromaticity. Typically when reading SMILES a tool will assign a Kekulé form and then re-aromatise.
SOME FALSE NEGATIVES

Kekulise

Aromatise