



THE SECRETS OF FAST SMARTS MATCHING

John Mayfield and Roger Sayle

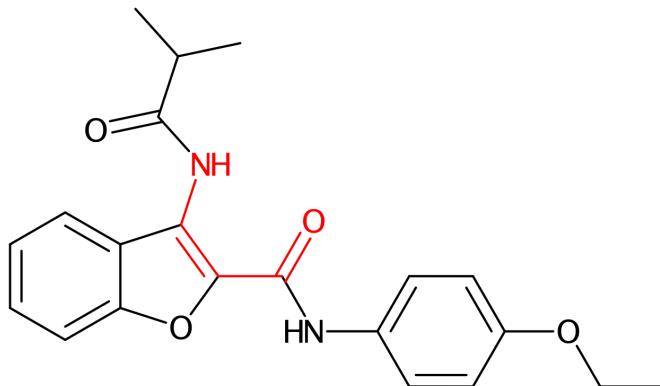
NextMove Software Ltd



SMARTS AND SUBSTRUCTURE SEARCH

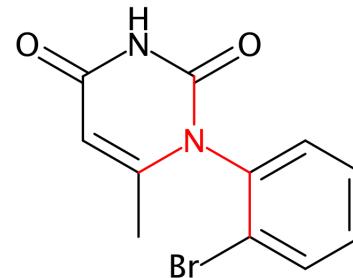
SMARTS is a specialisation of **substructure search** that allows complex atom and bond expressions

- Fingerprint prescreening can be less effective
- More dependence on backtracking search efficiency



O=[C,N]aa[N,O;!H0]

intramolecular H Bonds



[aD3]a!@-a([aD3])[aD3]

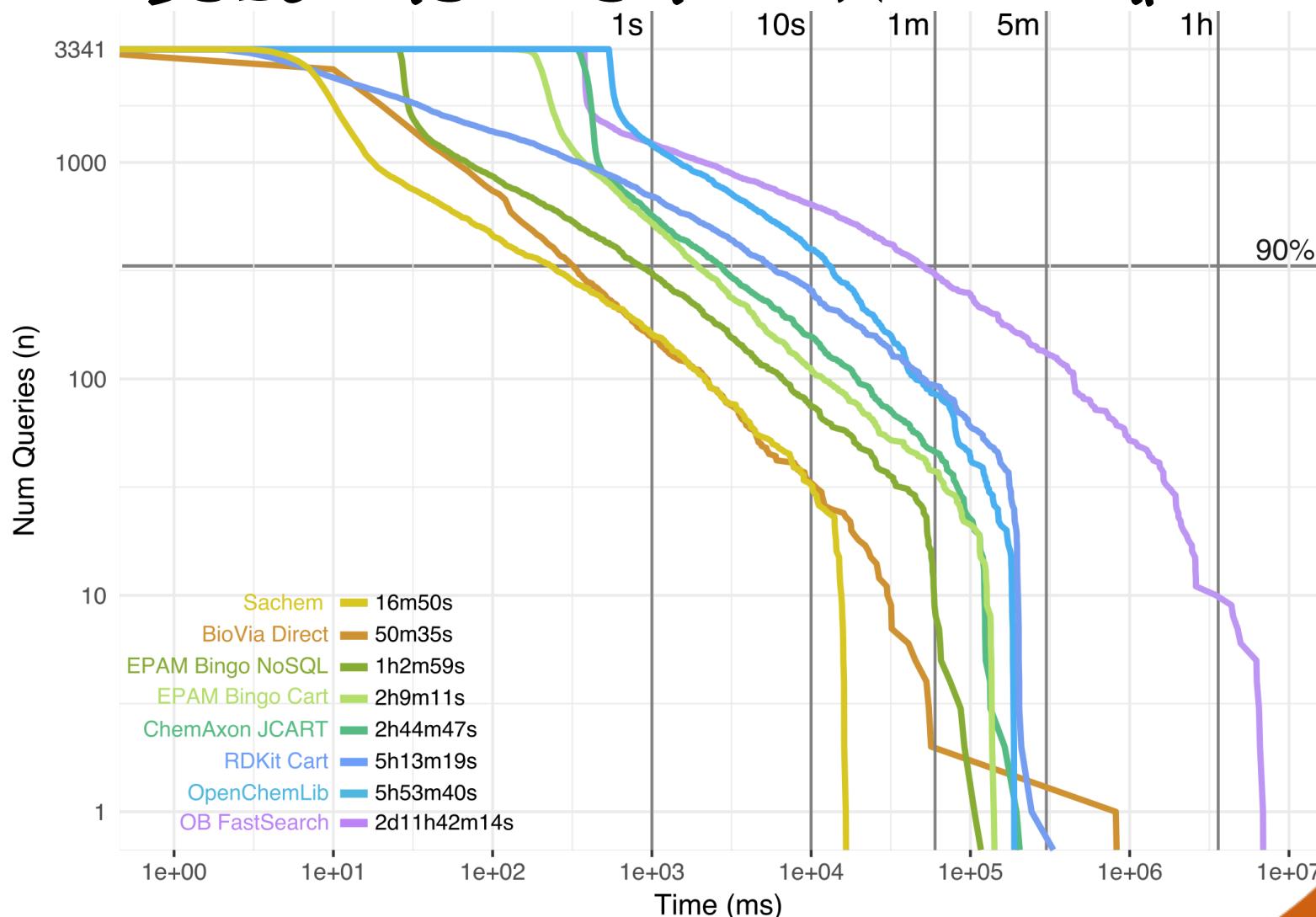
biaryl atropisomers

R. Sayle. Improved SMILES Substructure Searching. Daylight MUG. 2000.

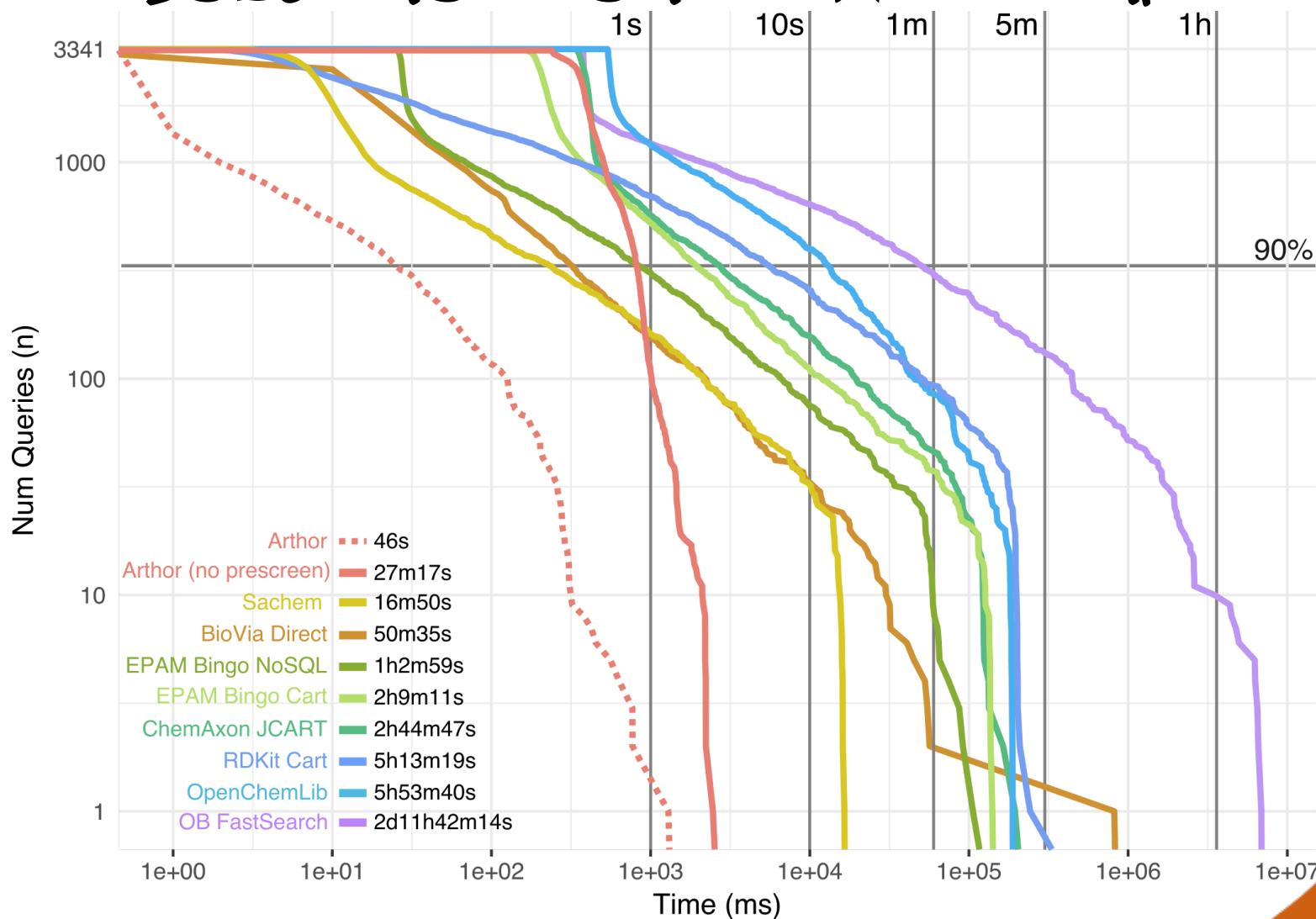
<https://www.daylight.com/meetings/emug00/Sayle/substruct.html>



SUBSTRUCTURE BENCHMARK



SUBSTRUCTURE BENCHMARK



May J. 2015. https://www.nextmovesoftware.com/talks/May_SubsearchBenchmark_201505.pdf



MORE POWER!

Running SMARTS over large structure sets can be **slow**.

Hilbig and Rarey (2015) report **42h26m** to check **1.172M** compounds for the PAINS patterns.

With the right tools we can check a datasets **~100x** larger in **32s**:

```
$ time ./atdbgrep -j 16 wehi_pains_rdkit.sma pubchem.smi.atdb -s  
pubchem.smi > pubchem_pains.smi
```

```
real 0m31.228s  
user 5m54.840s  
sys 0m11.727s
```



TO VF2 AND BEYOND!



THE FAVOURITE

VF2 2001

Linear memory

Prune with terminal sets

VF2 Open Source Chemistry Toolkit Implementations

RDKit

Code/GraphMol/Substruct/vf2.hpp

OpenBabel

src/isomorphism.cpp

Chemistry Development Kit

org.openscience.cdk.isomorphism.VentoFoggia

Ehrlich and Rarey *Journal of Cheminformatics* 2012, **4**:13
<http://www.jcheminf.com/content/4/1/13>



RESEARCH ARTICLE

Open Access

Systematic benchmark of substructure search in molecular graphs - From Ullmann to VF2

Hans-Christian Ehrlich and Matthias Rarey*

- L. Cordella, et al, A (sub)graph isomorphism algorithm for matching large graphs, *IEEE Transactions on Pattern Analysis and Machine Intelligence* Volume 26 Issue 10, p1367-1372, 2004.



THE FAVOURITE

VF2 2001

Linear memory

Prune with terminal sets

VF2 Plus 2015

Query traversal ordering

Sub-linear candidate selection

VF2++ 2018

Improved query traversal ordering

Prune with adjacent label counts

VF2 Open Source Chemistry Toolkit Implementations

RDKit

Code/GraphMol/Substruct/vf2.hpp

OpenBabel

src/isomorphism.cpp

Chemistry Development Kit

org.openscience.cdk.isomorphism.VentoFoggia

Ehrlich and Rarey *Journal of Cheminformatics* 2012, 4:13
<http://www.jcheminf.com/content/4/1/13>



RESEARCH ARTICLE

Open Access

Systematic benchmark of substructure search in molecular graphs - From Ullmann to VF2

Hans-Christian Ehrlich and Matthias Rarey*

Bespoke algorithms in other toolkits show higher memory usage, not covered in this talk.

- A. Jüttner and P. Madaras. VF2++—An improved subgraph isomorphism algorithm. *Discrete Applied Mathematics*, 2018, Volume 242, p69-81
- V. Carletti *et al.* VF2 Plus: An Improved version of VF2 for Biological Graphs. Graph-Based Representations in Pattern Recognition. GbRPR 2015. *Lecture Notes in Computer Science*, Volume 9069. Springer, Cham
- L. Cordella, *et al.*, A (sub)graph isomorphism algorithm for matching large graphs, *IEEE Transactions on Pattern Analysis and Machine Intelligence* Volume 26 Issue 10, p1367-1372, 2004.



VF2 IMPLEMENTATION

```
boolean match() {
    if (nMapped == nQryAtoms)
        return true;
    Pair pair = new Pair(); // pair := (u, v)
    while (nextPair(pair)) { // 1. select candidate pair, O(|nMolAtoms|)
        if (feasible(pair)) { // 2. check labels, adj, prune, O(deg(u)*deg(v))
            add(pair); // 3. map, update Tq/Tm, O(deg(u)+deg(v))
            if (match())
                return true;
            remove(pair); // 4. recurse
        }
    }
    return false;
}
```

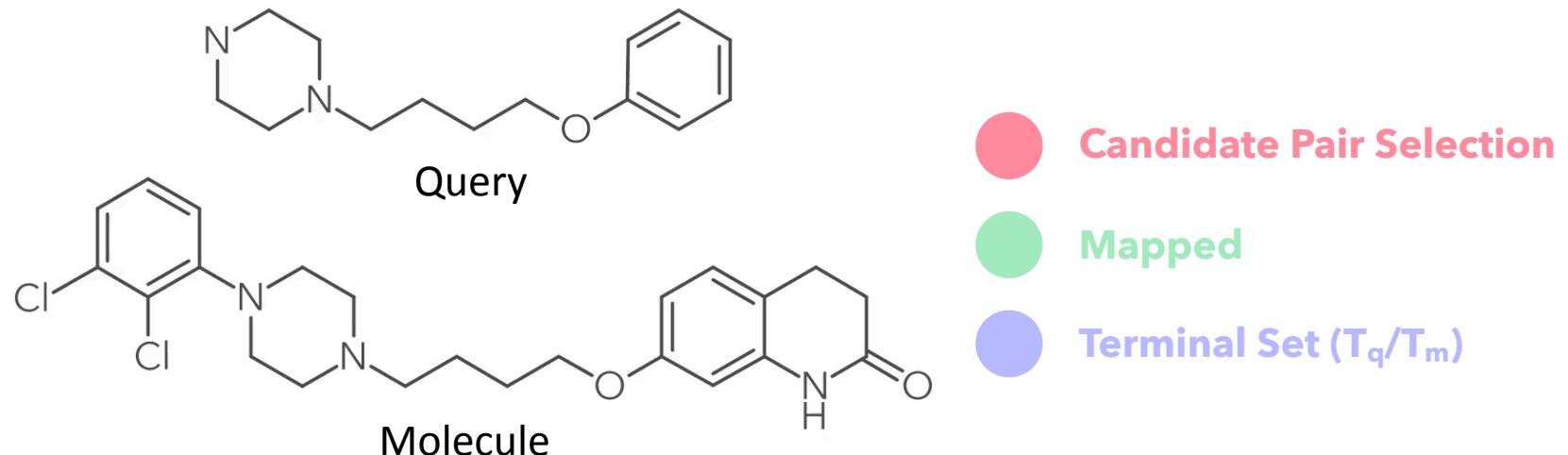
Maintain two terminal sets (T_q/T_m) of atoms adjacent to mapped atoms

- Candidates selected from this set, but takes linear time!
- Comparing candidate adjacency into of T_q and T_m allows pruning

Where q = Query (the substructure to be found), m = Molecule (the structure being matched), u and v is a candidate pair, u from the query, v from the molecule.
 $\deg(u)$ is the degree (number of bonds) of u



VF2 FOR CHEMICAL GRAPHS



VF2Plus improves candidate selection but still re-checks adjacency relations $O(\deg(u) \deg(v))$.

Terminal set pruning weaker for **non-induced¹** subgraph isomorphism

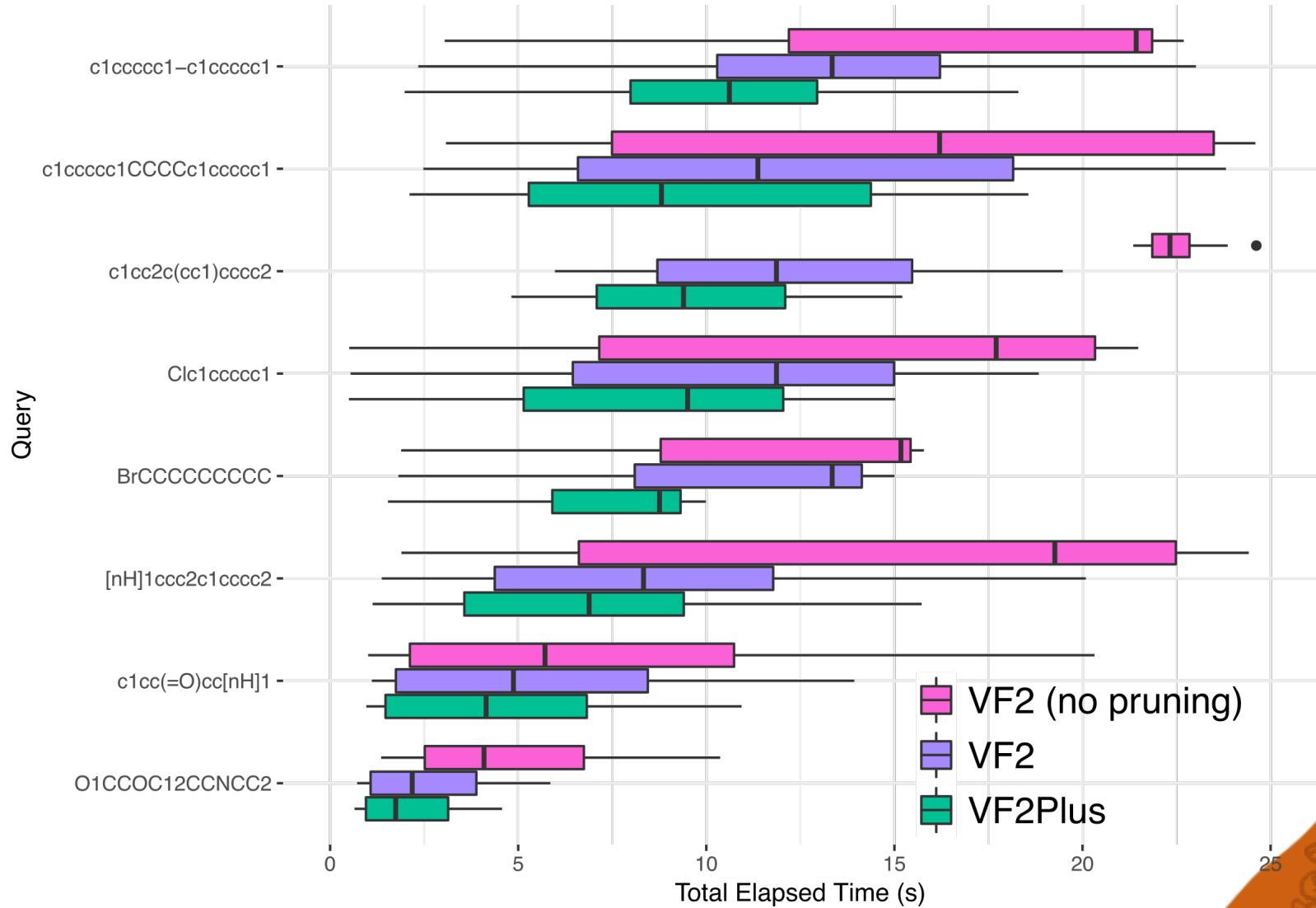
- Additional bookkeeping of this information
- Simple degree bound $\deg(qryAtom) \leq \deg(molAtom)$ does better

¹induced means hexane is not substructure of cyclohexane, we don't want this!

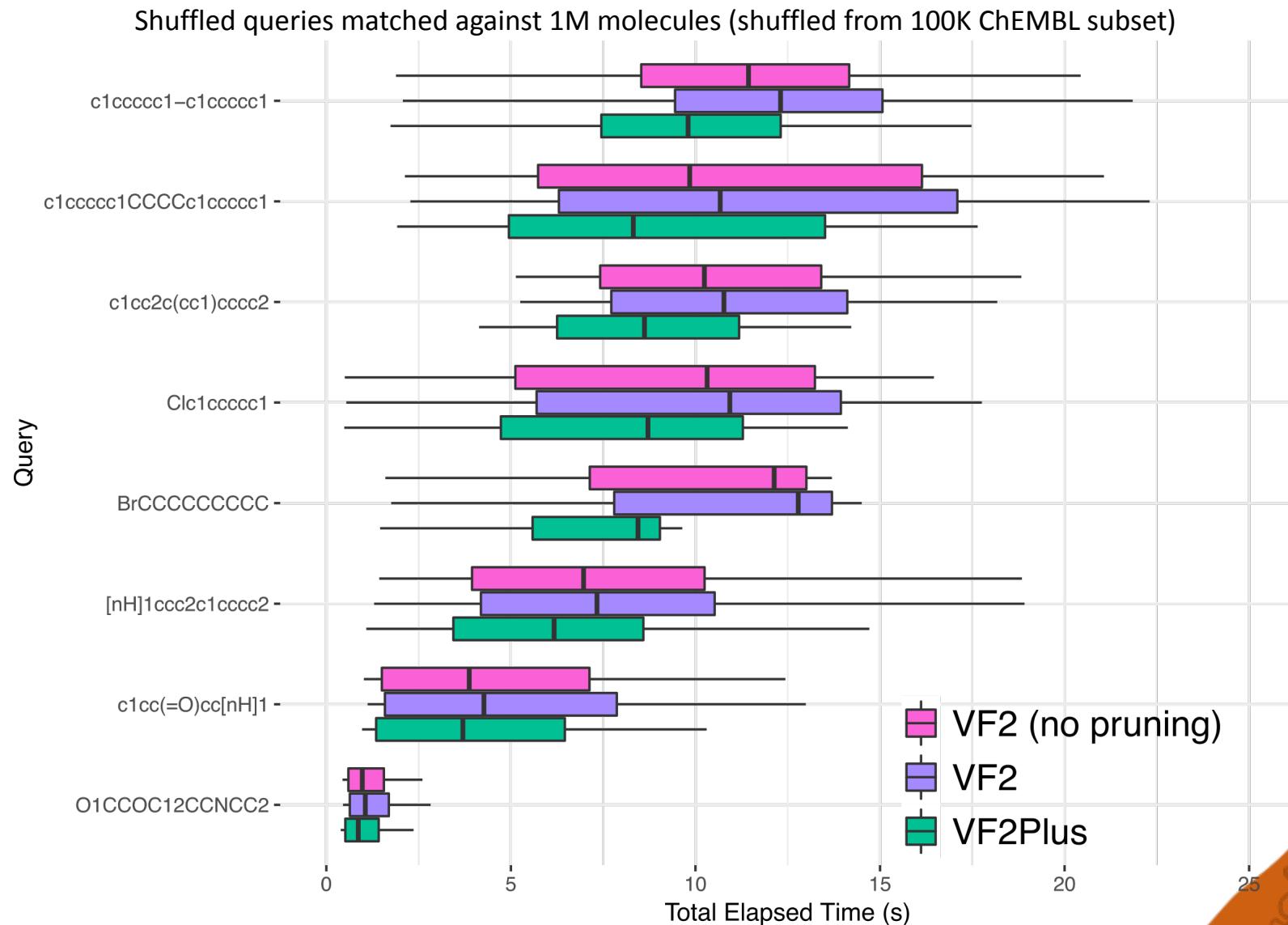


VF2 MATCH TIME

Shuffled queries matched against 1M molecules (shuffled from 100K ChEMBL subset)



VF2 MATCH TIME: $\text{deg}(\text{qryAtm}) \leq \text{deg}(\text{molAtm})$



THE RK ALGORITHM

Eliminate adjacency re-check of **VF2Plus** in feasibility function

Backtracking tree search selecting candidate query **bond** rather than **atom**

- R Sayle contributed **parsmart.cpp** to OELib (now Open Babel) **2001**
- AMBIT Smarts (**2011**) uses a similar algorithm citing Ray and Kirsch (**1957**)

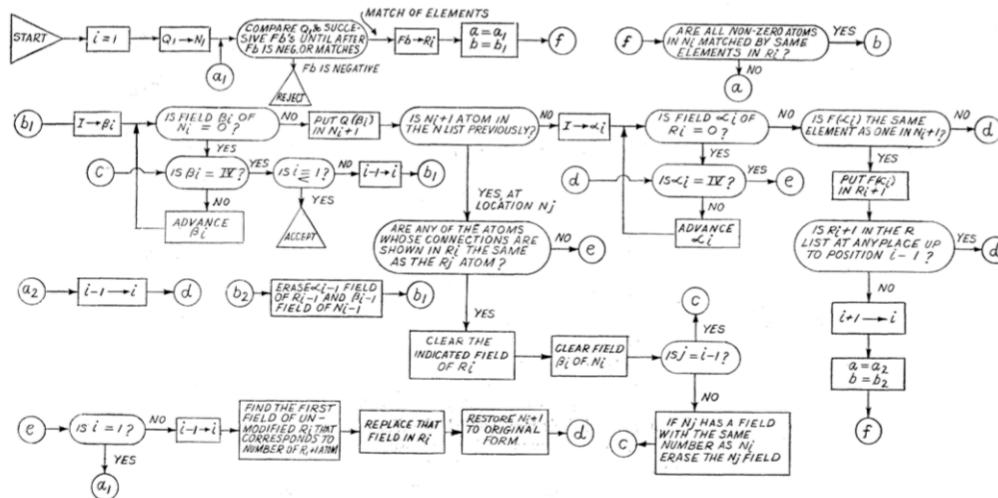


Fig. 5. Flow chart of the chemical-structure search routine.



RK IMPLEMENTATION

```
boolean match(int bndIdx) {  
    if (bndIdx == nQryBonds)  
        return true;  
  
    // ... qryBeg/End from bond[bndIdx], molBeg/End mapping of these  
  
    if (mapped(qryBeg) && mapped(qryEnd)) {  
  
    } else if (mapped(qryBeg)) {  
  
        beg and end are atoms at either end of a bond  
        qryBeg in a query bond and molBeg in the mol bond  
  
    } else {  
        // 1. seed atom, O(n)  
  
        for (Atom mAtm : mol.atoms()) {  
            if (feasibleAtom(qryBeg, mAtm)) {  
                add(qryBeg, mAtm);  
                if (match(bndIdx))  
                    return true;  
                remove(qryBeg);  
            }  
        }  
    }  
    return false;  
}
```



```

boolean match(int bndIdx) {
    if (bndIdx == nQryBonds)
        return true;

    // ... qryBeg/End from bond[bndIdx], molBeg/End mapping of these

    if (mapped(qryBeg) && mapped(qryEnd)) {

    } else if (mapped(qryBeg)) {           // 2. grow bond, O(deg(molBeg))

        for (Bond molBnd : molBeg.bonds()) {
            molEnd = molBnd.getNbr(molBeg);
            if (feasibleAtom(qryEnd, molEnd) &&
                feasibleBond(qryBnd, molBnd)) {
                add(qryEnd, molEnd);
                if (match(bndIdx + 1))          // recurse
                    return true;
                remove(qryEnd);
            }
        }
    } else {                                // 1. seed atom, O(n)

        for (Atom mAtm : mol.atoms()) {
            if (feasibleAtom(qryBeg, mAtm)) {
                add(qryBeg, mAtm);
                if (match(bndIdx))          // recurse
                    return true;
                remove(qryBeg);
            }
        }
    }
    return false;
}

```

RK IMPLEMENTATION

*beg and end are atoms at either end of a bond
 qryBeg in a query bond and molBeg in the mol bond*



```

boolean match(int bndIdx) {
    if (bndIdx == nQryBonds)
        return true;

    // ... qryBeg/End from bond[bndIdx], molBeg/End mapping of these

    if (mapped(qryBeg) && mapped(qryEnd)) {    // 3. close ring, O(deg(molBeg))

        Bond molBnd = molBeg.getBond(molEnd);
        if (feasibleBond(qryBnd, mbnd))
            return match(bndIdx + 1);                // recurse

    } else if (mapped(qryBeg)) {                    // 2. grow bond, O(deg(molBeg))

        for (Bond molBnd : molBeg.bonds()) {
            molEnd = molBnd.getNbr(molBeg);
            if (feasibleAtom(qryEnd, molEnd) &&
                feasibleBond(qryBnd, molBnd)) {
                add(qryEnd, molEnd);
                if (match(bndIdx + 1))           // recurse
                    return true;
                remove(qryEnd);
            }
        }
    } else {                                         // 1. seed atom, O(n)

        for (Atom mAtm : mol.atoms()) {
            if (feasibleAtom(qryBeg, mAtm)) {
                add(qryBeg, mAtm);
                if (match(bndIdx))           // recurse
                    return true;
                remove(qryBeg);
            }
        }
    }
    return false;
}

```

RK IMPLEMENTATION

*beg and end are atoms at either end of a bond
 qryBeg in a query bond and molBeg in the mol bond*



VIRTUAL MACHINE OP-CODES

We can replace the **if-then-else** with a **switch** on a set of op-codes:

```
boolean match(int bndIdx) {  
    if (bndIdx == nQryBonds)  
        return true;  
    if (mapped(qryBeg) && mapped(qryEnd)) {  
        // close ring  
        // ...  
    } else if (mapped(qryBeg)) {  
        // grow bond  
        // ...  
    } else {  
        // seed atom  
        // ...  
    }  
    return false;  
}
```

```
boolean match(int i) {  
    if (i == numOps)  
        return true;  
    switch (ops[i]) {  
        case CLOSERING:  
            // ...  
            break;  
        case GROWBOND:  
            // ...  
            break;  
        case SEEDATOM:  
            // ...  
            break;  
    }  
    return false;  
}
```

More efficient and enables additional logic!

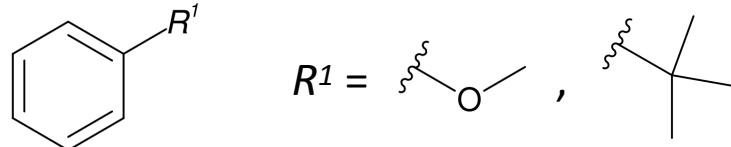


ADDITIONAL OP-CODES

A few of the additional operators we have:

SAMEPART	Two atoms are in the same connect component?
DIFFPART	Two atoms are in a different connect components?
RXNROLE	Atom is in a reactant/product/agent?
TETRA_LEFT/RIGHT	Tetrahedral stereo of atom is left/right?

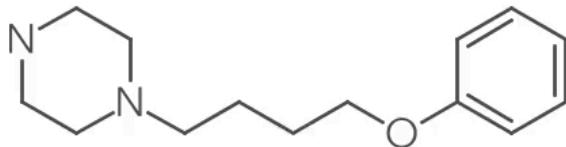
Stereochemistry and component grouping queries **([O]).([Na])** vs **([O].[Na])** are typically handled once all atoms are matched. These op-codes allows us to check these invariants ASAP.



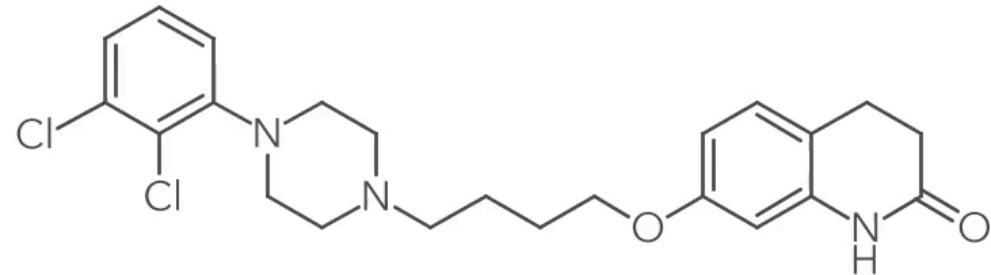
Recursive queries are handled by inlining sub-expression op-codes and introducing branches (**if-then-else**):



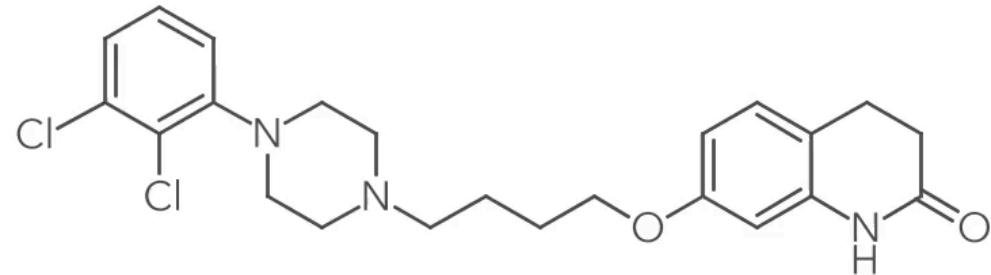
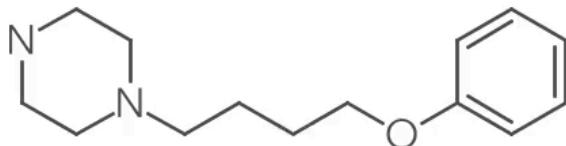
QUERY



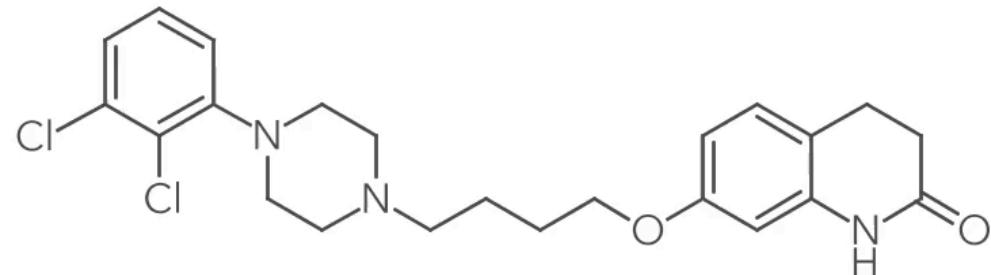
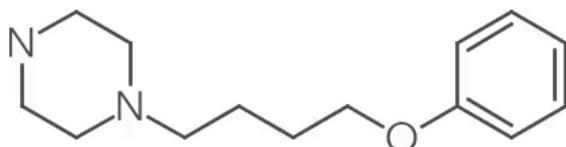
MOLECULE



VF2 Matching



VF2Plus Matching

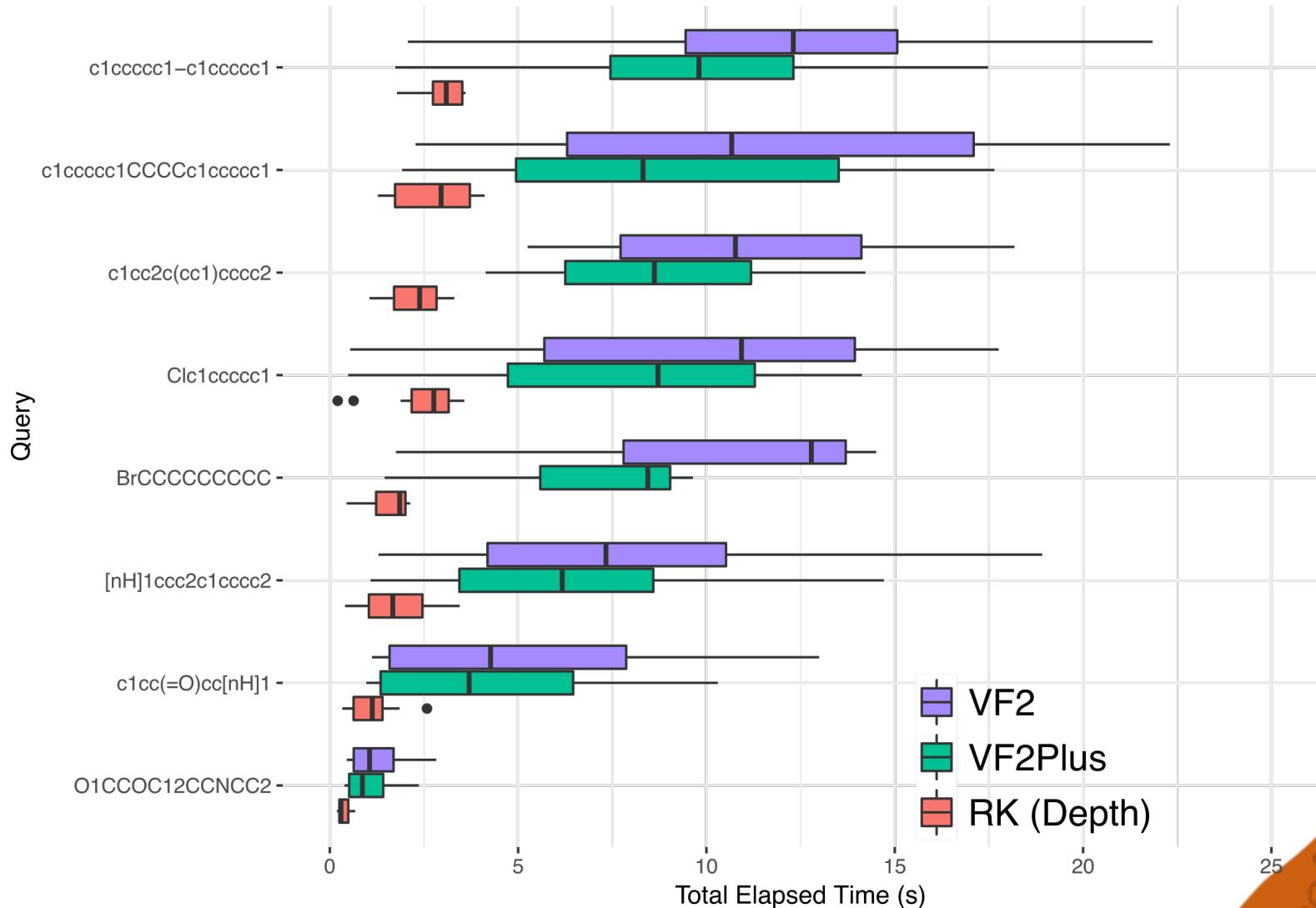


RK Matching



RK VS VF2

Shuffled queries matched against 1M molecules (shuffled from 100K ChEMBL subset)



QUERY TRAVERSAL ORDER



BEST-FIRST ORDERING

A query can be traversed (and matched) in a different orders

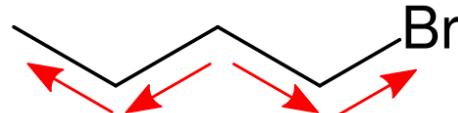
Try to choose a graph traversal order that is optimal

- Independent of **Ullmann**, **VF2**, or **RK**
- **VF2Plus** describes the general idea – select least probable first

1 . 0x CCCCCBr



1 . 6x CC (Br) CC



5 . 6x BrCCCC



SMARTS PROBABILITIES

The tricky part is knowing which expressions are least probable:

- Table driven **AtomFreq()**, **BondFreq()** and **AtomBondFreq()** functions
- Calculate the probability that a given SMARTS Atom/Bond expression matches a typical atom/bond

Estimate with care:

	AFreq (beg)	AFreq (end)	BFreq	ABFreq (end) Estimate	ABFreq (end) Actual
S=O	S: 0.9%	O: 11.46%	7.55%	0.86%	0.90%
S=C	S: 0.9%	C: 33.04%	7.55%	2.49%	0.61%

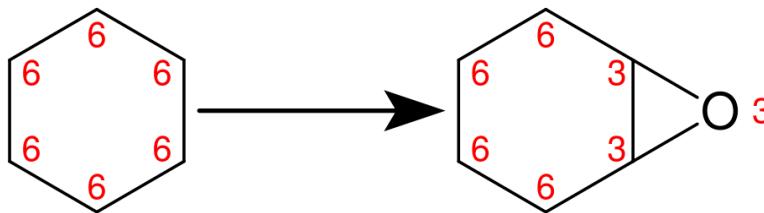
Choosing a **good seed atom** gets you most of the way:

- Any hetero atom, if no hetero atoms choose highest degree

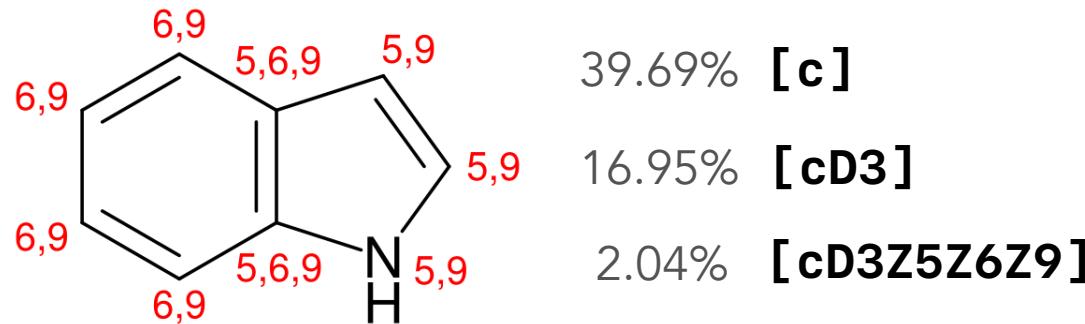


RING SIZE INVARIANTS

Smallest Ring Size (e.g. [r6]) is not invariant in a subgraph:



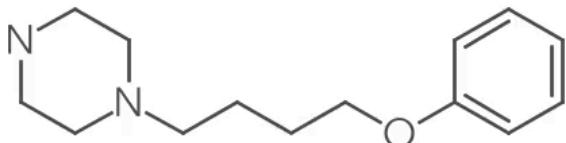
However Ring Size is! Daylight planned to introduce the [Z<n>] operator:



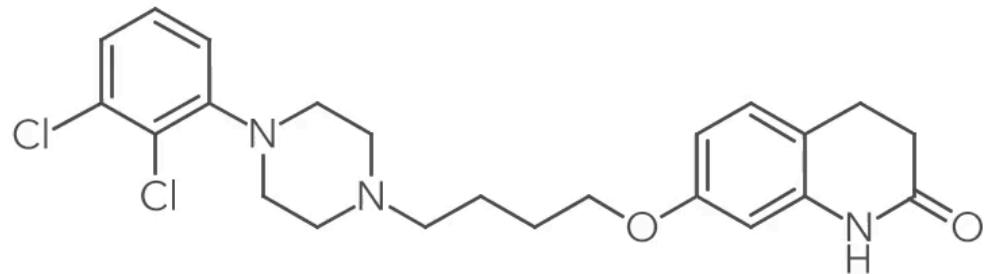
Improved probability estimates and additional pruning.



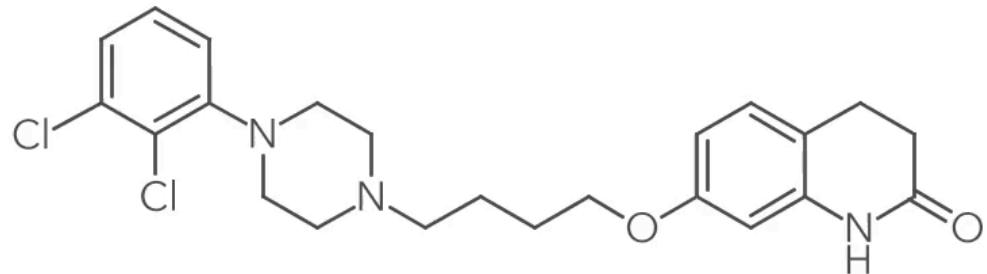
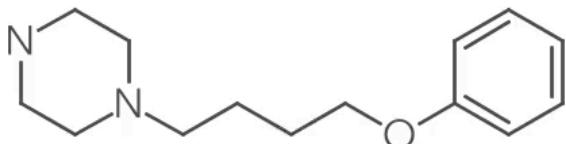
QUERY



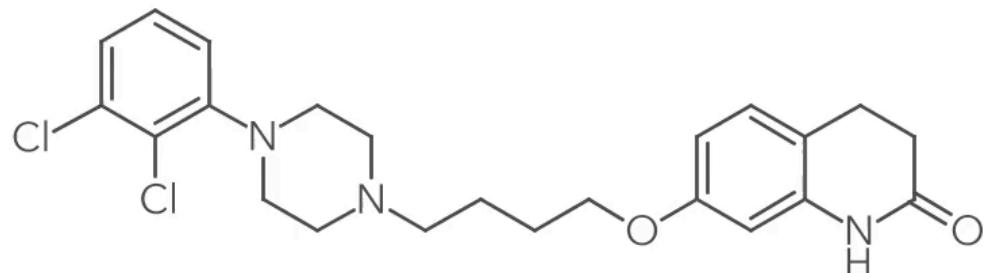
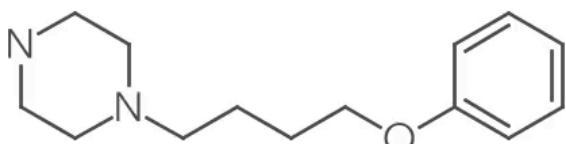
MOLECULE



Depth-First Matching



Breath-First Matching

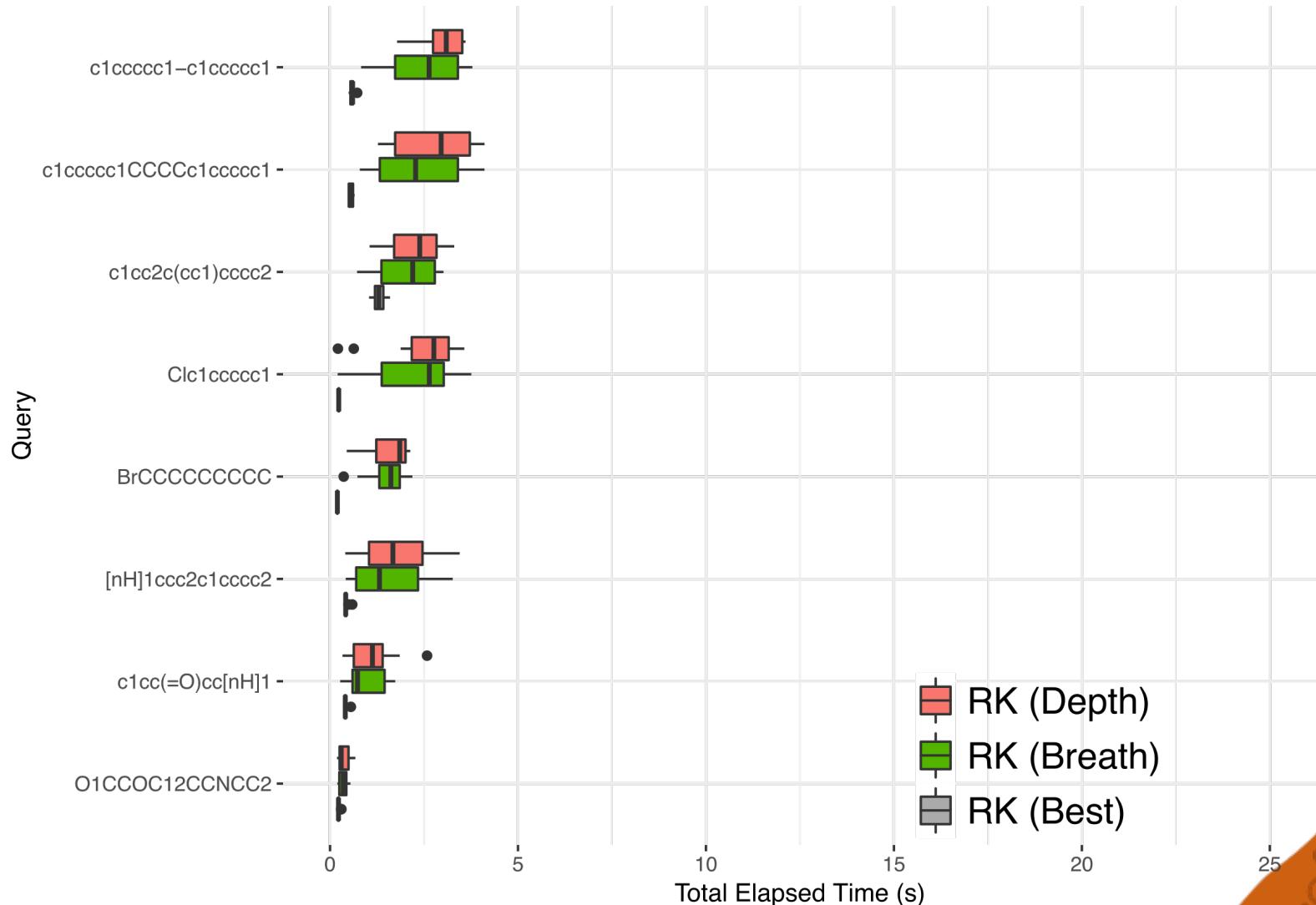


Best-First Matching



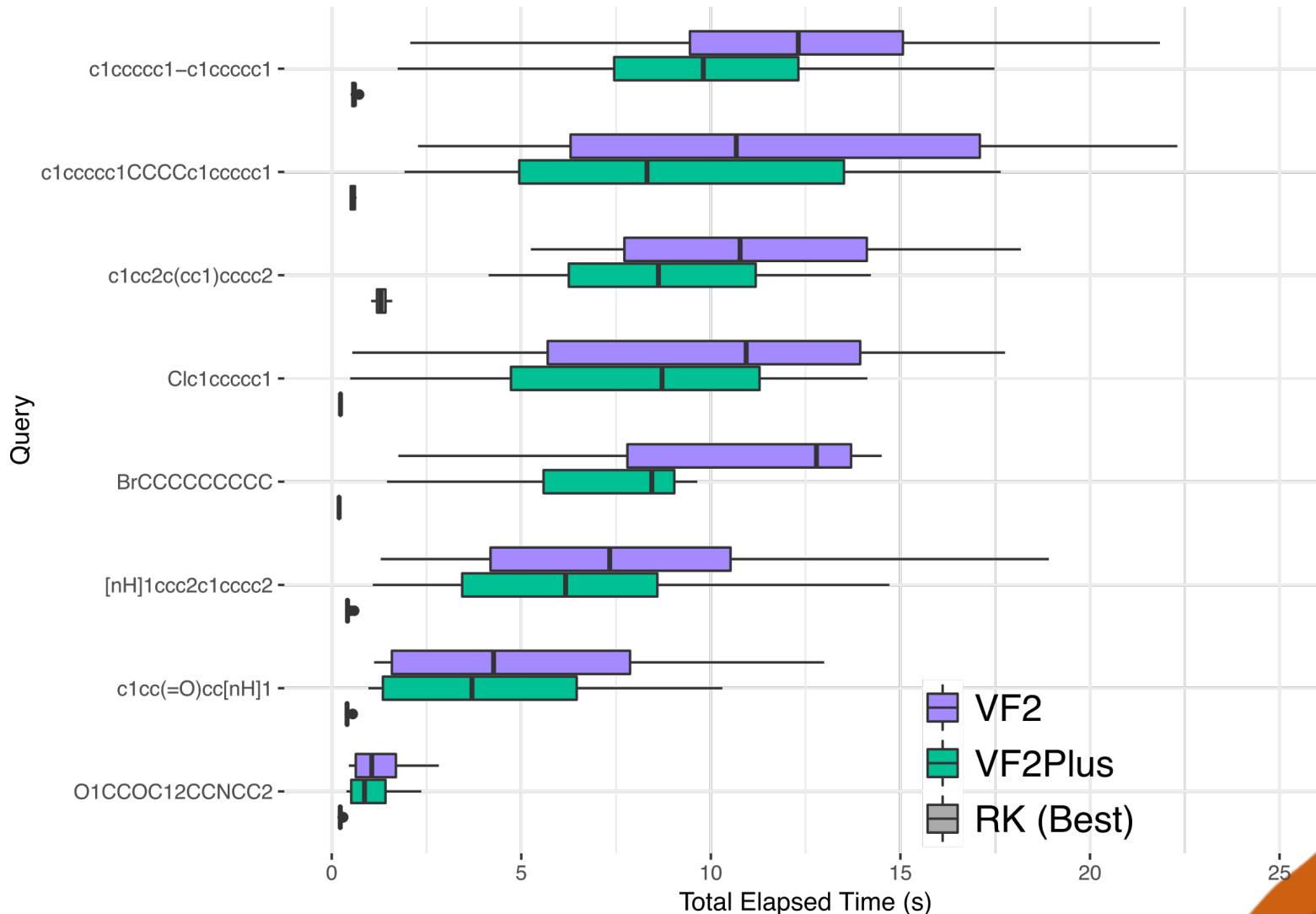
DIFFERENT TRAVERSAL MATCH TIMES

Shuffled queries matched against 1M molecules (shuffled from 100K ChEMBL subset)



RK (BEST) VS VF2

Shuffled queries matched against 1M molecules (shuffled from 100K ChEMBL subset)



ATDB FILE-FORMAT



MOST MOLECULES ARE BORING 1

The **majority of molecules** can be described by a **small number** of local atom environments ($r=0$)

For the purposes of SMARTS matching our **atom types** consist of the **primitive** properties we want to match:

- Atomic Number
- Formal Charge
- Total Hydrogen Count
- Implicit Hydrogen Count
- Degree
- Valence
- Ring Bound Count
- etc...

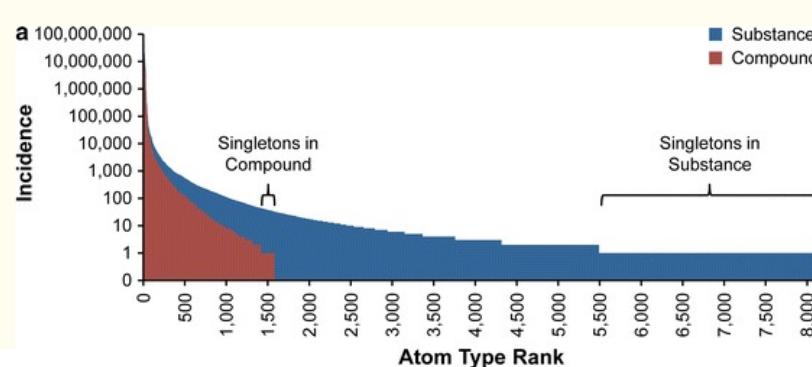


Fig 5a: Hähnke *et al.* PubChem atom environments. *J Cheminform.* 2015. 7 (41)



ATDB-FILE GENERATION

1. Compute frequency of atom types for a given database (**first pass**)
 2. Order atom types by their frequency, and assign a rank to each
 3. Write two types of ATDB records (**second pass**):
 - 1 byte for common atom-types ($0 \leq \text{rank} \leq 255$)
 - 2 bytes for rarer atom types ($256 \leq \text{rank} \leq 65535$)
-
- PostgreSQL cartridge does a less optimal single-pass possible by storing atom type table separately
 - Example two-pass indexing times:
 - **6 secs** for ChEMBL 25 (~2M mols)
 - **40 mins** for Zinc 2018-08-13 (~1B mols)



MOST MOLECULES ARE BORING 2

	Num Mols	Num Atom Types	% Mol 1 Byte Per Atom	Avg Bytes Per Mol
ChEMBL 25	1,870,461	490	99.44%	170
PubChem Compound¹	95,945,555	4357	99.29%	138
PubChem Substance¹	209,606,360	18,886	99.02%	133
Enamine REAL 2017	336,985,301	86	100.00%	117
Enamine REAL 2018	719,205,875	83	100.00%	124
Zinc²	939,613,211	285	99.999997%	116

¹PubChem Compound and Substance download on June 2018. ²Zinc downloaded on October 2018



SUMMARY

Conclusions

- **VF2** pruning less useful for chemical graphs
- **VF2Plus** improves candidate selection but not used
- **RK** performs best on chemical graphs
- **Op-code** based matcher expands query possibilities
- **Traversal order** optimised using probabilities
- **Co-design** of ATDB file-format for optimal matching, compression and minimise memory allocation

Future Work

- Always more optimisations
- Publication in preparation
- RDKit Patch

Acknowledgements

Noel O'Boyle

Richard Gowers



ARTHOR

Not Secure | internal.nextmovesoftware.com/arthur/ John

Arthor Search Manage Datasets Version 1.3

Structure Input Matched 14,575,833 results in 243 ms

Enter SMILES, Name, Identifier...

Q Similarity Substructure

ChEMBL 23

Enamine REAL [337M, Q3.4 2017]

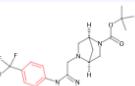
eMolecules [Jan 2018]

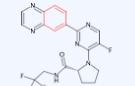
PubChem Compound [95M, June 2018]

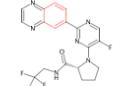
ChEMBL 24

ChemSpace Building Blocks [46M, June 2018]

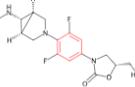
USPTO Exemplified Compounds

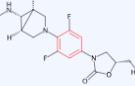
7,627,045  US09750719B2 Intermediate 17
C₁₉H₂₅F₃N₅O₂
410.495

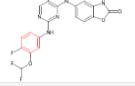
7,627,046  US08686143B2 Compound 165
C₁₉H₂₆F₄N₆O
417.433

7,627,047  US20120157429A1 Compound 165
C₁₉H₂₆F₄N₆O
417.433

7,627,048  US20140221336A1 Compound 165
C₁₉H₂₆F₄N₆O
417.433

7,627,049  US06875784B2 Example 26
C₁₉H₂₆F₂N₄O₅
376.388

7,627,050  US20040127530A1 Example 26
C₁₉H₂₆F₂N₄O₅
376.388

7,627,051  US08324200B2 Compound I...
C₁₉H₁₈F₃N₅O₃
421.374

©2018 NextMove Software Ltd. All Rights Reserved. Page Response 13 ms

www.nextmovesoftware.com/arthur.html

