Supporting data

Lemon: a modern C++ tool for the rapid development of structural benchmarking datasets

Jonathan Fine[1], and Gaurav Chopra[1*]

[1]Department of Chemistry, Purdue University, 720 Clinic Drive, West Lafayette, Indiana, USA

Obtaining Lemon

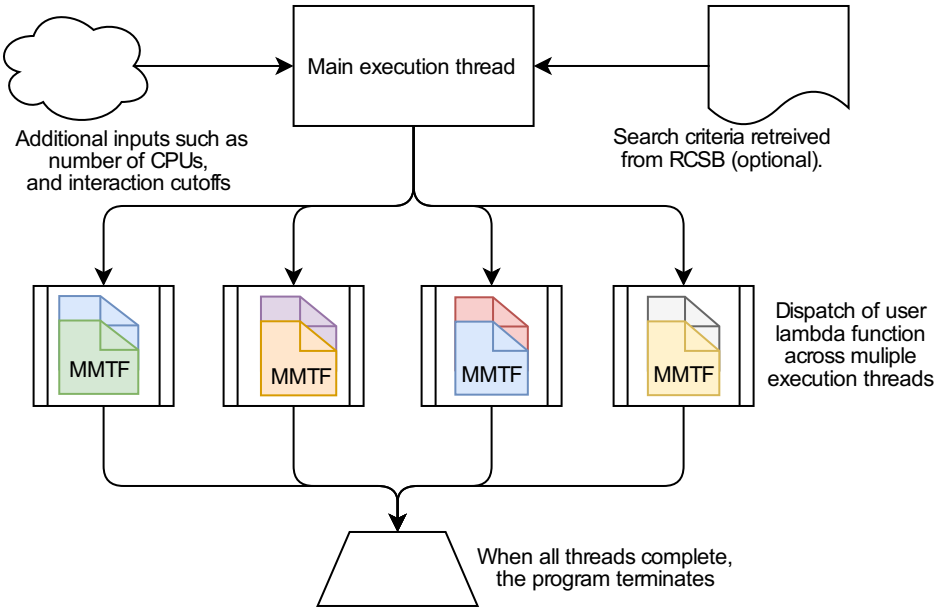
The lemon benchmarking framework is written in C++11. Its only dependencies are a C++11 compiler, the CMake tools, the Boost C++ libraries (specifically the 'filesystem' library, and the Chemfiles library). It should be noted that the Chemfiles library will be automatically installed by the lemon install scripts, if it is not explicitly installed by the user.

Lemon has been freely released on GitHub. To obtain the software, complete the following steps:

```
$ git clone https://github.com/chopralab/lemon.git
$ cd lemon/
$ mkdir build/
$ cd build
$ cmake .. -DCMAKE_BUILD_TYPE=Release
$ make
```

The example binaries will be created in the 'progs' subdirectory of `build`. It is recommended that you supply the 'make' command with an additional argument '-j ##' where '##' is the number of physical cores on your machine.

Figure S1: Flow chart showing the execution of a lemon program.



Example Lambda Functions

Listing S1: C++ Lambda function to count the number of biological assemblies in the PDB

```
auto worker = [](const chemfiles::Frame& complex, const std::string& pdbid) {

    // Desired info is obtained directly
    auto result = lemon::count_bioassemblies(complex);

    // Custom output phase
    std::stringstream ss;
    ss << pdbid << " " << result << "\n";
    std::cout << ss.str();
};
```

Listing S2: C++ Lambda function to count the number of residues in the PDB

```
// Map to hold individual residue counts in a thread safe manner
std::unordered_map<std::thread::id, lemon::ResidueNameCount> resn_counts;

auto worker = [&resn_counts](const chemfiles::Frame& complex,
                             const std::string& pdbid) {

    // Desired info is calculated directly, no pruning, output is done
    auto th = std::this_thread::get_id();
    lemon::count_residues(complex, resn_counts[th]);
};
```

Listing S3: C++ Lambda function to find metal ions in the PDB

```
auto worker = [](const chemfiles::Frame& complex, const std::string& pdbid) {

    // Selection phase
    auto result = lemon::select_metal_ions(complex);

    // No pruning, straight to output phase
    lemon::print_residue_name_counts(std::cout, pdbid, complex, result);
};
```

Listing S4: C++ Lambda function to count all the small molecules in the PDB

```
auto worker = [](const chemfiles::Frame& complex, const std::string& pdbid) {

    // Selection phase
    auto smallm = lemon::select_small_molecules(complex);

    // Pruning phase
    lemon::remove_identical_residues(complex, smallm);
    lemon::remove_cofactors(complex, smallm, lemon::common_cofactors);
    lemon::remove_cofactors(complex, smallm, lemon::linear_molecules);

    // Output phase
    lemon::print_residue_name_counts(std::cout, pdbid, complex, smallm);
};
```

Listing S5: C++ Lambda function to count the number of small molecules which interact with a peptide within a distance cutoff.

```
auto worker = [dist_cutoff](const chemfiles::Frame& complex,
                            const std::string& pdbid) {

    // Selection phase
    auto peptides = lemon::select_peptides(complex);
    auto smallm = lemon::select_small_molecules(complex);

    // Pruning phase
    lemon::remove_identical_residues(complex, smallm);
    lemon::remove_cofactors(complex, smallm, lemon::common_cofactors);
    lemon::remove_cofactors(complex, smallm, lemon::linear_molecules);
    lemon::keep_interactions(complex, smallm, peptides, dist_cutoff);

    // Output phase
    lemon::print_residue_name_counts(std::cout, pdbid, complex, smallm);
};
```

Listing S6: C++ Lambda function to count the number of small molecules which interact with a nucleic acid within a distance cutoff.

```
auto worker = [dist_cutoff](const chemfiles::Frame& complex,
                            const std::string& pdbid) {

    // Selection phase
    auto nucleic_acids = lemon::select_nucleic_acids(complex);
    auto smallm = lemon::select_small_molecules(complex);

    // Pruning phase
    lemon::remove_identical_residues(complex, smallm);
    lemon::remove_cofactors(complex, smallm, lemon::common_cofactors);
    lemon::remove_cofactors(complex, smallm, lemon::linear_molecules);
    lemon::keep_interactions(complex, smallm, nucleic_acids, dist_cutoff);

    // Output phase
    lemon::print_residue_name_counts(std::cout, pdbid, complex, smallm);
};
```

Listing S7: C++ Lambda function to count the number of small molecules which interact with a metal ion within a distance cutoff.

```
auto worker = [dist_cutoff](const chemfiles::Frame& complex,
                            const std::string& pdbid) {

    // Selection phase
    auto metals = lemon::select_metal_ions(complex);
    auto smallm = lemon::select_small_molecules(complex);

    // Pruning phase
    lemon::remove_identical_residues(complex, smallm);
    lemon::remove_cofactors(complex, smallm, lemon::common_cofactors);
    lemon::remove_cofactors(complex, smallm, lemon::linear_molecules);
    lemon::keep_interactions(complex, smallm, metals, dist_cutoff);

    // Output phase
    lemon::print_residue_name_counts(std::cout, pdbid, complex, smallm);
};
```

Listing S8: C++ Lambda function to count the number of small molecules which interact with a Heme group within a distance cutoff.

```
auto worker = [dist_cutoff](const chemfiles::Frame& complex,
                            const std::string& pdbid) {

    // Selection phase
    auto hemegs = lemon::select_specific_residues(complex,
        {"HEM", "HEA", "HEB", "HEC"}
    );
    auto smallm = lemon::select_small_molecules(complex);

    // Pruning phase
    lemon::remove_identical_residues(complex, smallm);
    lemon::remove_cofactors(complex, smallm, lemon::common_cofactors);
    lemon::remove_cofactors(complex, smallm, lemon::linear_molecules);

    lemon::keep_interactions(complex, smallm, hemegs, dist_cutoff);

    // Output phase
    lemon::print_residue_name_counts(std::cout, pdbid, complex, smallm);
};
```

Listing S9: C++ Lambda function to count the number of small molecules which interact with a SAM molecule within a distance cutoff.

```
auto worker = [dist_cutoff](const chemfiles::Frame& complex,
                            const std::string& pdbid) {

    // Selection phase
    auto sam = lemon::select_specific_residues(complex, {"SAM"});
    auto smallm = lemon::select_small_molecules(complex);

    // Pruning phase
    lemon::remove_identical_residues(complex, smallm);
    lemon::remove_cofactors(complex, smallm, lemon::common_cofactors);
    lemon::remove_cofactors(complex, smallm, lemon::linear_molecules);

    lemon::keep_interactions(complex, smallm, sam, dist_cutoff);

    // Output phase
    lemon::print_residue_name_counts(std::cout, pdbid, complex, smallm);
};
```

Listing S10: C++ Lambda function to count the number of small molecules which do not interact with any water molecules within a distance cutoff.

```
auto worker = [dist_cutoff](const chemfiles::Frame& complex,
                            const std::string& pdbid) {

    // Selection phase
    auto waters = lemon::select_specific_residues(complex, {"HOH"});
    auto smallm = lemon::select_small_molecules(complex);

    // Pruning phase
    lemon::remove_identical_residues(complex, smallm);
    lemon::remove_cofactors(complex, smallm, lemon::common_cofactors);
    lemon::remove_cofactors(complex, smallm, lemon::linear_molecules);

    lemon::remove_interactions(complex, smallm, waters, dist_cutoff);

    // Output phase
    lemon::print_residue_name_counts(std::cout, pdbid, complex, smallm);
};
```

Table S1: Common residues ignored in the above examples.

Three letter code	Full name
CIT	Citric Acid
FLC	Citric Acid Ion
ICT	Isocitrate
SAM	S-Adenosylmethionine
SIA	O-Sialic Acid
CLA	Chlorophyll
HEM	PROTOPORPHYRIN IX CONTAINING FE
HEA	Heme-A
HEB	Heme-B/C
HEC	Heme-C
MES	2-(N-MORPHOLINO)-ETHANESULFONIC ACID
EPE	4-(2-HYDROXYETHYL)-1-PIPERAZINE ETHANESULFONIC ACID
GOL	Glycerol
FAD	FLAVIN-ADENINE DINUCLEOTIDE
FMN	FLAVIN MONONUCLEOTIDE
NAD	NICOTINAMIDE-ADENINE-DINUCLEOTIDE
NAP	NADP NICOTINAMIDE-ADENINE-DINUCLEOTIDE PHOSPHATE
ADP	ADENOSINE-5'-DIPHOSPHATE
ATP	ADENOSINE-5'-TRIPHOSPHATE
GDP	GUANOSINE-5'-DIPHOSPHATE
GTP	GUANOSINE-5'-TRIPHOSPHATE
DTP	2'-DEOXYADENOSINE 5'-TRIPHOSPHATE
BE7	(4-CARBOXYPHENYL)(CHLORO)MERCURY
MHA	(CARBAMOYLMETHYL-CARBOXYMETHYL-AMINO)-ACETIC ACID
DHD	2,4-DIOXO-PENTANEDIOIC ACID
B3P	2-[3-(2-HYDROXY-1,1-DIHYDROXYMETHYL-ETHYLAMINO)-PROPYLAMINO]-2-HYDROXYMETHYL-PROPANE-1,3-DIOL
BTB	2-[BIS-(2-HYDROXY-ETHYL)-AMINO]-2-HYDROXYMETHYL-PROPANE-1,3-DIOL
NHE	2-[N-CYCLOHEXYLAMINO]ETHANE SULFONIC ACID

Table S2: Linear residues ignored in the above examples.

Three letter code	Full name
PG4	TETRAETHYLENE GLYCOL
PG5	1-METHOXY-2-[2-(2-METHOXY-ETHOXY]-ETHANE
PG6	1-(2-METHOXY-ETHOXY)-2-{2-[2-(2-METHOXY-ETHOXY]-ETHOXY}-ETHANE
1PE	PENTAETHYLENE GLYCOL
2PE	NONAETHYLENE GLYCOL
1PG	2-(2-{2-[2-(2-METHOXY-ETHOXY)-ETHOXY]-ETHOXY}-ETHOXY)-ETHANOL
PE4	2-{2-[2-(2-{2-[2-(2-ETHOXY-ETHOXY)-ETHOXY]-ETHOXY}-ETHOXY)-ETHOXY]-ETHOXY}-ETHANOL
PE8	3,6,9,12,15,18,21-HEPTAOXATRICOSANE-1,23-DIOL
PG6	1-(2-METHOXY-ETHOXY)-2-{2-[2-(2-METHOXY-ETHOXY]-ETHOXY}-ETHANE
P33	3,6,9,12,15,18-HEXAOXAICOSANE-1,20-DIOL
C8E	(HYDROXYETHYLOXY)TRI(ETHYLOXY)OCTANE
XPE	3,6,9,12,15,18,21,24,27-NONAOXANONACOSANE-1,29-DIOL
N8E	3,6,9,12,15-PENTAOXATRICOSAN-1-OL
DR6	3,6,9,12,15-PENTAOXATRICOSAN-1-OL
MYR	MYRISTIC ACID
OTE	2-{2-[2-(2-OCTYLOXY-ETHOXY)-ETHOXYL]-ETHOXY}ETHANOL
OLA	OLEIC ACID
OLB	(2S)-2,3-dihydroxypropyl (9Z)-octadec-9-enoate
OLC	(2R)-2,3-dihydroxypropyl (9Z)-octadec-9-enoate
PLM	PALMITIC ACID
PAM	PALMITOLEIC ACID
PEE	1,2-Dioleoyl-sn-glycero-3-phosphoethanolamine
LHG	1,2-DIPALMITOYL-PHOSPHATIDYL-GLYCEROLE
MC3	1,2-DIMYRISTOYL-RAC-GLYCERO-3-PHOSPHOCHOLINE
SPM	SPERMINE
SPK	SPERMINE (FULLY PROTONATED FORM)
SPD	SPERMIDINE

Table S3: Timings for the listed lambda function when tested on the entire PDB. Each run was performed with 8 threads enabled. A distance cutoff of 6 was used for all programs which require this parameter. The PDB archive was retrieved on July 17th, 2018.

Program	Runtime in seconds
Count biological residues	1315
Count all residues	1246
Count all metal ions	1301
Count all small molecules	1260
Count all small molecules that interact with protein	1844
Count all small molecules that interact with RNA/DNA	1356
Count all small molecules that interact with metal	1288
Count all small molecules that interact with a Heme	1282
Count all small molecules that interact with SAM	1287
Count all small molecules that do not interact with water	1289
Average	1346.8

Figure S2: Histogram showing the number of bioassemblies assigned to a given crystal structure. Note that no assemblies are assigned to NMR structures.

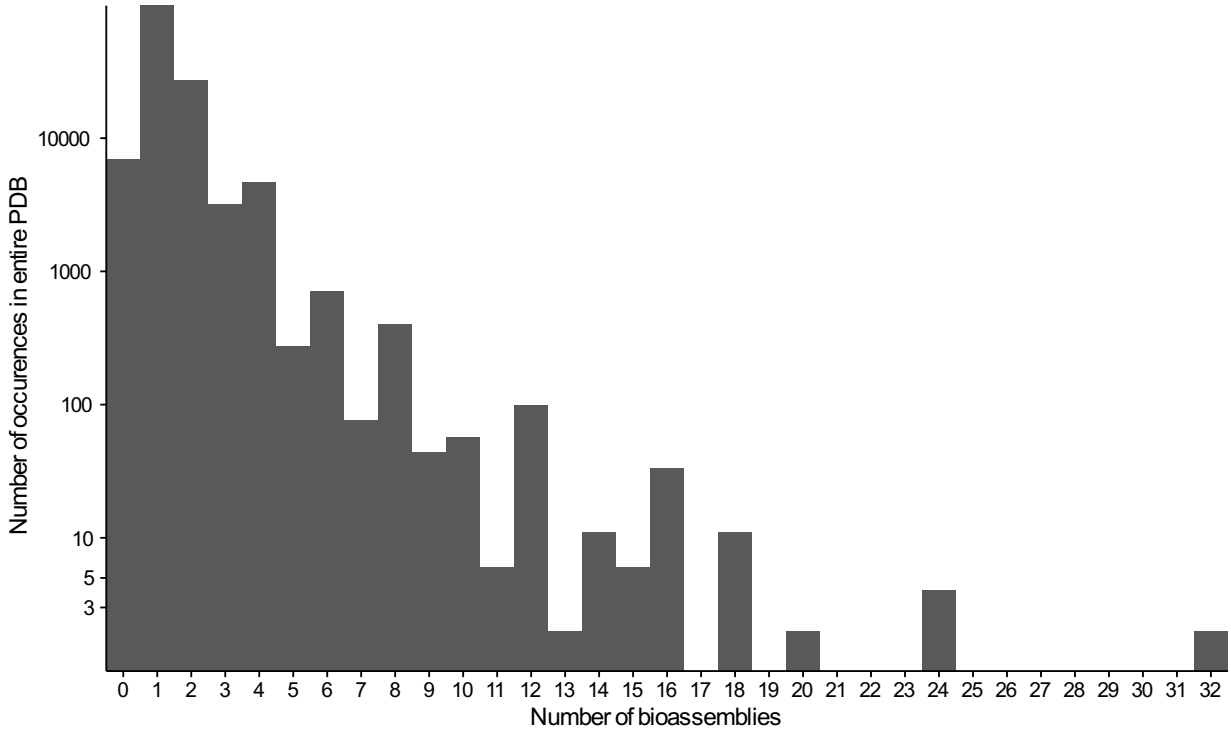


Figure S2: Histogram showing the number of times a unique residue is repeated (maximum of 250 repetitions). The X-axis is the number of times a residue is repeated. For example, if the residue '409' occurs once in PDBID 142N and thrice in PDBID 1L59, it is 'repeated' four times. The y-axis gives the frequency repetition.

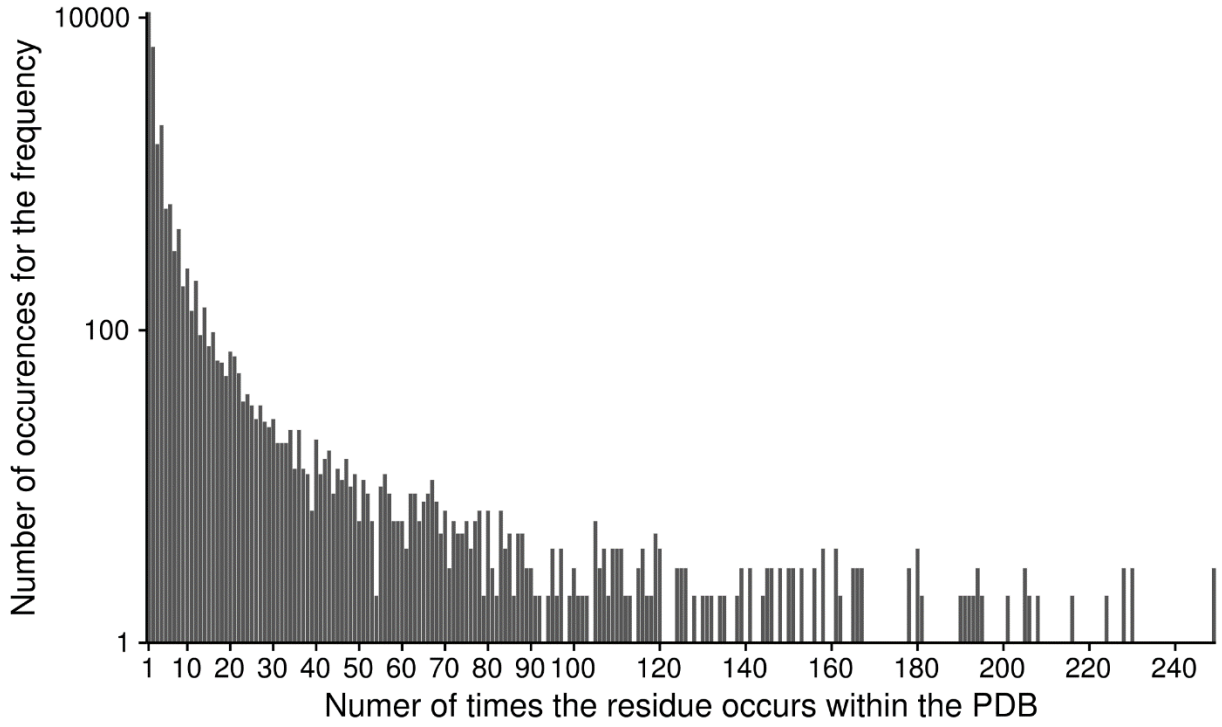


Table S4: Number of PDB entries which contain a given metal ion.

Metal Element	Number of entries
Li	72
Be	1 (4P4K)
Na	7537
Mg	13418
K	2435
Ca	10197
V	1 (1QYL)
Mn	3138
Fe	2175
Cu	1432
Hg	477

Table S5: Compounds which interact with SAM.

PDBID	Three letter code	Count
1P7L	PPK	2
5KJK	6T1	1
5KJM	6TM	1
5KJN	6TL	1
4E47	0N6	1
2G70	HNT	1
2G72	F21	1
1R30	DTB	2
1RG9	PPK	4
2ZVJ	KOM	1
3A7E	DNC	1
5LSA	DNC	1
3PFG	TLO	1
1H1D	BIA	1
1VID	DNC	1
3BWM	DNC	1
3BWY	DNC	1
3BXO	UPP	2
3S68	TCW	1
3S7B	NH5	1
4JDS	1L4	1
4JLG	1L8	1
3DMH	GMP	1
1XDS	DRA	2
4KIC	PPY	2
2PXC	G3A	1
3EMB	GTG	1
4M7T	25W	1
2BR4	P4C	3
2CL5	BIE	1
4NDN	PPK	2
3I5U	5NA	2
4NJG	HHS	2
4NJH	2K8	2
4NJI	2K8	2
4NJJ	2K8	2
4NJK	2KA	2
4A6E	ASE	1
4ODJ	3PO	1

5T8S	3PO	1
4RVG	TYD	1
5UL4	B12	1
5V37	8WD	1
5V3H	8WG	1
5W8A	A1S	1
5WBV	9ZY	1
4X61	3XV	1
4XUC	43G	1
4XUD	43H	1
4XUE	43J	1
4YND	4GQ	1
5A1I	AND, PPK	1, 1
5ARF	I9H	1
5ARG	H41	1
5AYF	C7H	1
5CCL	4ZW	1
5CCM	4ZX	1
5CPR	539	1
5EML	5QK	1
5FEP	41K	1
5FES	9SE	1
5FHQ	DNC	1
5FHR	DNC	2